博　士　論　文

グローバル時代における
ソフトウェア品質マネジメントの枠組みの
再構築と品質マネジメント強化手段に関する研究

2003年1月

保田　勝通

目次

第１章　はじめに

1.1　ソフトウェア品質マネジメント

ソフトウェアの品質を考えさせた最近の事例といえば，「西暦２０００年問題※」であろう．この問題は，いわゆる通常のソフトウェアバグとは趣を異にしている．設計者はメモリの節約のためあるいは慣習に従って，意図的に年号を下２桁で処理したのであり，当時の考え方としては当然であった．何が誤算であったかといえば，まさか２０００年までそのソフトウェアが使われるとは思っていなかったことであり，よしんば使われている場合でも簡単に直せると考えたことである．

この問題を冒頭で取り上げたのは，我々が漠然と思っているよりもはるかにいろいろなところでかなり長期間にわたってソフトウェアが使われているということであり，またその保守がいかに困難な作業かということである．

もう１つ直近の事例は，某銀行で合併後のシステムが正常に動作せず，膨大な取引の決済が遅延し，連日新聞紙上で大きく取り上げられ，銀行業績にも大きな影響を及ぼしただけでなく，社会的事件になった事例である．この事件はソフトウェアの品質が悪いというような技術的・管理的レベルというよりは，経営トップが，銀行の基幹業務の開発実態を把握せず，かつその影響を理解していなかったためといわれている．

この２つの象徴的な出来事は，ソフトウェア品質を無視しては，社会活動は円滑にできなくなっているということである．社会として，組織として，このような事態を招かないために，ソフトウェア品質マネジメントが必要であることの証左である．

従来，品質管理（Quality Control，あるいは略してQC）という用語が使われていたが，Control は管理というよりは制御に近く，実体に合わず適切でないという指摘がされてきた．そこで，従来総合的品質管理に対するTQC(Total Quality Control）という呼称を，1996 年に Total　Quality　Management　(TQM)に変更した．

ちなみに，TQM とは，「企業・組織における経営の“質”向上に貢献する管理技術，経営手法」である．デミング賞委員会でのTQM の定義は，次のとおりである．

※「西暦２０００年問題」とは，西暦２０００年になったときに一部のコンピュータシステム，コンピュータ製品，時刻処理を組み込んだマイコン部品等を使用している設備機器などが，年号を下２桁で処理しているために２０００年を１９００年と誤って動作することが原因で起こる，様々な問題のことをいう．

「顧客の満足する品質を兼ね備えた品物やサービスを適時に適切な価格で満足できるように，企業の全組織を効果的・効率的に運営し，企業目的の達成に貢献する体系的活動」（「デミング賞のしおり」から）．

そこで，本論文では，品質管理の管理という用語を「マネジメント」と置き換え，品質マネジメントと呼ぶことにした．ちなみに，ＴＱＭ宣言が出された1996年の頃は，本論文が指摘する従来のソフトウェア開発が混乱し，品質面においても何をどうすべきか議論をしていた時期である．

さて，本論文のタイトルに品質マネジメントという用語を配したのは，上記のＴＱＭの定義の観点から，ソフトウェアの品質を捉えようという立場を示すためである．ここで，品質の代表的な定義を紹介しておく[1]．

まず，ＪＩＳやＩＳＯ規格では，「品質とは，ユーザの要求を満足させるために製品のもつべき特性である」と規定している．一方，品質管理の観点からは，クロスビー(P.B. Crosby) [37]が「品質とは，要求に対する適合である」と定義したのが，おそらく社会的に認められた最初の定義である．その後，日本では石川馨先生が「品質はユーザの満足度である」と定義され[2]，これが1980年代世界に冠たる日本の品質管理隆盛の原動力になったのである．その後，米国が日本を研究し，1990年代米国の繁栄を導いた一因であるMB賞（マルコム・ボルドリッジ国家品質賞）のスローガンとなった「Customer Satisfaction(CS:顧客満足度)」が有名になったが，これはいわば日本への逆輸入である．

さて，ソフトウェアの分野では，「ソフトウェア品質特性」というＩＳＯ規格が制定されている．ここでは，機能性，信頼性，使用性，効率性，保守性，移植性という６つの品質特性が定義されており，その下に多岐にわたる２１の副特性があるという構造である[3]．

本論文のソフトウェア品質の捉え方は，もちろん不良の多い少ないというような狭義の意味だけではなく，上記の「ソフトウェア品質特性」を踏まえた「品質はユーザの満足度」あるいは顧客満足度(CS)である．これを，コンピュータメーカやソフトウェアハウスなどのソフトウェアを開発し提供する，企業の視点で考えると，品質だけを切り離して考えてもあまり意味がない．少なくともプロジェクトを成功させなければ，企業経営上，問題となる．実際問題として，品質，コスト，納期という従来のプロジェクト管理の３つの要素は密接にからんでいることが普通である．

したがって，本論文の第6章以降の個別強化施策では，あくまでも品質という視点に立ちながら，その施策の対象としては，ソフトウェア開発プロセス全般にわたり，プロジェクトマネジメントの対象範囲全体を視野に入れている．具体的には，従来，プロジェクトマネジメントの分野で弱点と言われた「リスクマネジメント」から，各開発プロセス対応の施策のみならず，ソフトウェア工学教育までを対象にしている．そのおのおのの施策が，ソフトウェア品質マネジメントとどう関わっているのかは，各章で説明する．

1.2 本研究の背景

日本は 1980 年代に，高度成長のもと，その高い生産性と高品質で世界を席捲し，日本の「ソフトウェア工場」[4]システムも注目を浴びた．筆者は，自ら経験してきた，主として１９８０年代までの日立製作所のソフトウェア品質保証活動を集大成し，1995 年に「ソフトウェア品質保証の考え方と実際」[1]という著書を上梓した．これが本研究の基礎となっている．

しかしながら，1990 年代には，日本経済のバブル崩壊が進行するなかで，情報産業は，「ネオダマ」といわれるネットワーク化，オープン化，ダウンサイジング化，マルチメディア化という大変革の時代を迎え，ソフトウェア開発現場は混乱した．こうした状況を分析し新しい方向性を見出そうとする動きは，社内外で起こってきた．本研究は，筆者が入社以来携わってきた日立製作所内でのソフトウェア品質管理活動の経験と，社外での継続的な研究活動や標準化活動を通して得られた知見を集大成したものによるところが大きい．

本研究は，筆者の日立におけるソフトウェア品質保証および生産技術を中心とする業務に密着した部分と，日本科学技術連盟に設置されたソフトウェア生産管理研究委員会での品質管理研究推進活動や，ISO/IEC JTC1 情報技術国際標準化対応の情報処理学会学会／情報規格調査会／SC7(ソフトウェアエンジニアリング)，日本規格協会での ISO 9000 ソフトウェア品質保証に関する標準化活動[38]という，この 10 数年間の社外委員会活動を通じて得られた部分からの両面の成果である．

従来，品質問題については，品質保証部門の施策が主体であった．しかし，近年は，より広い視点，すなわち上流工程の設計部門での「品質の作り込み」が重視され，また QCD を主体とする古典的なプロジェクト管理より幅広い視点をもつ，モダンプロジェクトマネジメントやエンタープライズプロジェクトマネジメント[5]が注目されている．こうした状況を踏まえ，本論文での研究内容は以下の 2 点に重点を置いている．１つは，受注活動段階から開始されるリスクマネジメントやプロジェクトオフィスに代表されるモダンプロジェクトマネジメントの取り込みとその改善に代表される広義の「ソフトウェア品質マネジメントの枠組み」である．２つ目は，ソフトウェア品質マネジメントの視点に立ち，上流開発工程に重点を置いた，ソフトウェアエンジニアリングと開発管理の融合による，具体的なソフトウェア品質の向上施策に関する研究である．

1.3 本研究の目的と取り組み

本研究の最終目標は，品質マネジメントの視点に立った，事業への貢献と顧客満足度の向上であり，ひいては社会への貢献である．そのためには，本研究のタ

イトルである「グローバル時代におけるソフトウェア品質マネジメントの枠組みの再構築と品質マネジメント強化手段に関する研究」が必要である．そこで，ようやくソフトウェア開発現場が落ち着きを取り戻しつつある現時点で，前節で述べた状況下で開始した社内外の試行錯誤の試みを通じて得られたソフトウェア品質マネジメントに関する様々な知見を整理し体系化を試みた．

本研究の目的は，第１に，1980 年代に輝かしい成果を出した日本のソフトウェア開発が，1990 年代になぜ混乱し失速したのかを，主として品質マネジメントの視点から分析し，課題を整理する．第２に，この課題に対する「ソフトウェア品質マネジメントの枠組み再構築」の提案と，「品質マネジメント強化の具体的手段の研究およびその考察」を行うことである．

具体的な取り組みは，以下の通りである．

最初に 1980 年代までに日本の品質管理が達成した成果を確認する．次に，主として日本科学技術連盟のソフトウェア生産管理（ＳＰＣ）研究委員会での議論をベースとして，1990 年代のソフトウェア開発混乱の原因を分析し，21 世紀の課題を述べる．次に開発混乱の原因を集約し，この課題を整理する．さらに，これに対する日立製作所における研究と対策事例を示す．具体的には，まず「グローバル時代のソフトウェア品質マネジメント」の枠組みの提案を行う．次に日立製作所の情報システム部門として取り組んできた，開発工程別の対策事例に対応した研究内容を紹介する．

1.4　本論文の構成

本論文は全１３章から構成されており，後続する各章の概要は以下の通りである．

第 2 章では，1980 年代までに日本で確立された「ソフトウェア品質マネジメントの考え方」と，その後のグローバル化への鍵を握るソフトウェア開発管理の標準化動向を述べる．

第３章では，「1990年代日本のソフトウェア品質マネジメントの課題と取り組み方」ということで，これまでのソフトウェア品質管理の変遷を整理した上で，１９９０年代日本のソフトウェア開発混乱の原因分析を行う．これに基づき，１９９０年代日本の課題と対策の検討を行い，２１世紀へ向けての課題を整理する．

第 4 章では，「ソフトウェア品質マネジメントの枠組みの再構築と具体的強化手段の概要」ということで，１９９０年代日本のソフトウェア開発混乱の原因とそれに対する課題を踏まえて，ソフトウェア品質保証の課題に対する対策方針と，各対策の位置付けを述べる．すなわち，ここで，本論文の具体的研究テーマとその位置付けを明確にする．

第５章以降が，本論文の具体的研究内容である．

第 5 章では，「グローバル時代のソフトウェア品質マネジメント」の枠組みの

提案を行う.具体的には,「グローバルスタンダードと整合する品質マネジメントシステム」,および「品質マネジメントシステムを実現するための組織と仕組み」の提案を行い,合わせて「伝統的品質保証システムへの改良」についても述べる.

第 6 章から第 12 章は,各管理面・技術面からの具体的品質およびプロジェクトマネジメントに関する施策の提案および試行あるいは適用状況の報告である.前の第 5 章が品質マネジメントの枠組みの提案であるのに対して,この７つの章は,具体的施策の研究報告であり,論文構成上はこれらを１章にまとめたほうが美しいが,量的に内容が多いことと,本研究が品質マネジメントという広い範囲を総括的に扱っているため,各テーマがかなり独立しているので,それぞれ独立した章立てとした.

第6章から第12章までは,概ねソフトウェア開発工程に対応した施策である.

第６章「モダンプロジェクトマネジメントの適用」と第７章の「仕様確定状況の監視支援ー仕様発散防止ＱＦＤ」は,主に受注時までの上流工程の課題の対策である.

第８章「レビュー品質向上手段の導入による設計品質の向上」は設計段階の品質向上施策である.

第９章「プロジェクトナビゲーションツールによる管理精度の向上」は,開発段階全般を通してのプロジェクト管理精度向上施策であり,第１０章「ソフトウェアの品質評価精度の向上と効率的なフィードバック手法」は,主としてテスト段階での早期品質評価とフィードバックによる品質向上策である.

第１１章は,最近急増している内部仕様の分からない情報機器やソフトウェアパッケージ,それにネットワーク製品の組み合わせによる情報システムの品質保証手段としての「マルチベンダ構成の情報システム向けシステムシミュレーションテスト（ＳＳＴ)」を取り上げた.

第１２章では,組織におけるソフトウェア開発力の源泉である教育を取り上げ,「ソフトウェア技術者実践教育ＳＥＰによるスキルの継承と向上」を考察する.最後に,第 13 章では,本研究で得られた研究成果をまとめ,今後の課題と展望について述べる.

第2章　ソフトウェア品質マネジメントの考え方

2.1 はじめに

本章[6]の構成は次の通りである．2.2では，ソフトウェア開発の特質をハードウェアと対比して整理し，これまでに得られた知見やソフトウェア工学発展のあゆみを概観する．2.3 では，１９８０年代に確立された日本式ソフトウェア品質管理の特徴および代表的事例を紹介する．2.4 では，ソフトウェアのトラブル事例をまとめ，発注者・供給者双方の留意点を紹介する．2.5 では，前節の留意点に対応するソフトウェア品質向上施策を述べる．最初に開発のフェーズに対応した 品質保証施策を説明し，次に開発プロセスの改善，さらにノウハウの伝承という観点から米国のベストプラクティスを紹介する．2.6 では，最近活発に推進されているソフトウェア開発管理の標準化動向として，影響力の大きい２つのテーマ，すなわちソフトウェア・ライフサイクル・プロセス規格（ＳＬＣＰ），ソフトウェアの能力成熟度モデル（ＣＭＭ）に関する位置付けと概要を紹介する．

2.2 ソフトウェア開発の特質

信頼性工学で取り上げられてきた研究内容の大半は，物理法則に従うハードウェアに起因する事柄である．まず，物理法則に拠らないソフトウェア開発の特質は何かを考えてみよう．

（1）ソフトウェア開発の特質

ブルックス（F. P. Brooks）は，名著「人月の神話」 [7]の中でソフトウェア開発の特質として，複雑性，適応性，可変性，不可視性の４つを挙げている．

・複雑性：ソフトウェアは他の人工構造物より本質的に複雑で，規模の増加に従って複雑性は非線形に増大する．
・適応性：ソフトウェアは物理学のように従うべき統一原理があるのではなく，インタフェースを人間の習慣や社会制度，あるいは，ハードウェア等に従わせなければならない．
・可変性：ソフトウェアは，絶えず変化し続けるアプリケーションや利用者，慣習および機械・機器といった文化マトリクスにはめ込まれている．その変化がソフトウェアに変化を強制する．
・不可視性：ソフトウェアは，目に見えないものであり，視覚化できない．通常，

土地には地図，建築には間取り図，シリコンチップには回路図という具合に幾何学的抽象概念に基づく表現がある．ソフトウェア実体は，空間に埋め込めないため，その構造を図に表そうとすると，制御の流れ，データの流れ等複数の有向グラフで構成され，その構造は本質的に視覚化できないままである．

以上は，ソフトウェア開発の困難さを示す特質であるが，一方，有利な点もある．それは，ソフトウェアが物質で構成されていないため，物理的制約を受けず，経時的変化を受けないことである．ソフトウェアは，いじらない限り，劣化することはない．このことは，そのまま，高品質実現の困難さとその継続の有利さを示している．

（2）ソフトウェア開発に関する知見

ソフトウェア開発に関する知見[7]をいくつか紹介する．

●個人用プログラムの作成と，ソフトウェア製品の開発では，規模当たりの開発コストが数倍異なる．
●銀の弾丸（特効薬）はない．すなわち，「技術にせよ管理手法にせよ，単独で１０年間にソフトウェアの生産性や品質を飛躍的に改善するものはない．」
●ソフトウェア開発の成功の鍵は管理的側面にある．

ブルックスは，同著[7]のなかでソフトウェア開発の成功のためには，「プロジェクトに携わる人々の質，およびその組織形態と管理こそが，使用するツールや採用する技術的アプローチよりもはるかに重要な要因である」と述べている．これはもちろん，ソフトウェア開発の際，プログラミングや設計技法が必要ではないという意味ではなく，そういう技術さえあればソフトウェア開発はできるという誤解に対するコメントであり，その意味で筆者も同感である．

（3）ソフトウェア工学発展の歩み

約30年前にソフトウェア工学が提唱されて以来，その努力の大半はソフトウェアの特質である大規模かつ複雑な論理の集合体を正確に構築するための方法論の確立に費やされてきた．その成果が，構造化プログラミング，構造化設計法から，最近のオブジェクト指向分析・設計法等に現れている．こういう開発技法のような技術面からのアプローチに加えて，10年前頃からは，管理面を主体とする国際標準化活動が精力的に進められてきた．品質保証に関する国際規格ISO 9000のソフトウェアへの適用，ソフトウェア・ライフサイクル・プロセス（以降ＳＬＣＰと略す）の標準化，ソフトウェア開発組織に対する成熟度モデルとプロセス評価，プロジェクト管理（構成管理も含めて）のデファクト・ＩＳＯ標準のソフトウェアへの適用等が，その主なものである．

2.3　ソフトウェア品質管理の概要と特徴

本節では，１９８０年代に確立された日本式ソフトウェア品質管理の特徴および代表的事例を紹介する．

2.3.1 日本のソフトウェア品質管理の特徴[1]

（１）ハードウェア生産方式のアナロジー（ソフトウェア工場思想）

日本のソフトウェア開発を牽引したのは，コンピュータ開発を手がけていた電子通信製品の製造企業であった．彼らが，日本で大成功を収めつつある品質管理を取り入れてソフトウェア開発を早期に立ち上げようと考えたのは自然な成り行きである．これは取りも直さず，ソフトウェアを工業製品と考え，ハードウェア生産方式のアナロジーで開発することであり，ソフトウェア工場(software factory)思想に基づくと言われる所以である．

（２）ユーザ主導の要求仕様設定

先行した日本のメインフレームメーカが，ＯＳやミドルウェア等の汎用製品を開発するにあたっては，日本の製造業と同様，新製品のコンセプトを考案するというよりは，既に欧米にある製品仕様をカスタマイズしたり，その機能・性能を改良するという状況にあった．一方，大規模オンラインシステムのような個別受注製品の開発では，発注者である大企業が業務の詳細を承知しており，その機械化であるため，ソフトウェア開発者自身が製品コンセプトを作るニーズが少なかった．これらの要因により，開発者側の要求分析工程における製品コンセプト力(What)が育成されなかった．

（３）設計・製造工程(How)での品質重視（製品の完成度重視）

これも，他の工業製品と同様，日本では，汎用品であれ，特注品であれ，信頼性特性を主体とする品質を重視する．ソフトウェアについても同様である．この設計・製造工程での品質重視には，上記のソフトウェア工場制度は適していたため，その品質の高さは，世界的に見て高水準であった．

（４）日本的品質管理の導入（ＰＤＣＡ，ＱＣ７つ道具等の手法，全員参加，小集団活動）

これも日本社会のソフトウェア工場制度のもとでの組織的な開発に適した方法であった．いうまでもなく，管理のサイクルＰＤＣＡ　(Plan-Do-Check-Action)は戦後すぐ日本に品質管理を教えた宣教師ともいうべきデミング(W.E.Deming)博士により，もたらされた管理の進め方の原則である．日本では，これをＣＡＰＤとして運用した．つまり，開発プロセスでデータを取って問題を分析し(Check)，これを対策すると共に開発プロセスにフィードバックして「改善」するというのが日

本的品質管理の神髄ともいえるものである．日本ではソフトウェア開発の現場でも，個人主義の欧米と異なり，個人の品質データの収集分析はＱＣ７つ道具等の手法と相俟って普及し，全員参加による集団（チーム）活動としての高品質活動が欧米でも高く評価された．

2.3.2 代表的活動例

（1）基本ソフトウェア開発[1]

この分野は，日本の本格的ソフトウェア開発の草分けである．高度な技術が要求されるだけでなく，大規模であるため，プロジェクト管理特に品質管理が重視された．出荷権限を持つ独立した検査部門や，仕様書のデザインレビュー等上流工程の重視，プロセスの標準化，不良原因の追求等ソフトウェア品質管理の基礎が確立された．この領域が，ソフトウェア品質管理の成功領域である．日立製作所における品質保証がその代表例である．

図２．１は，出荷権限を持つ独立した品質保証部門が製品の検査と合否判定を行うだけではなく，上流工程から，品質管理に関与する活動を示している．

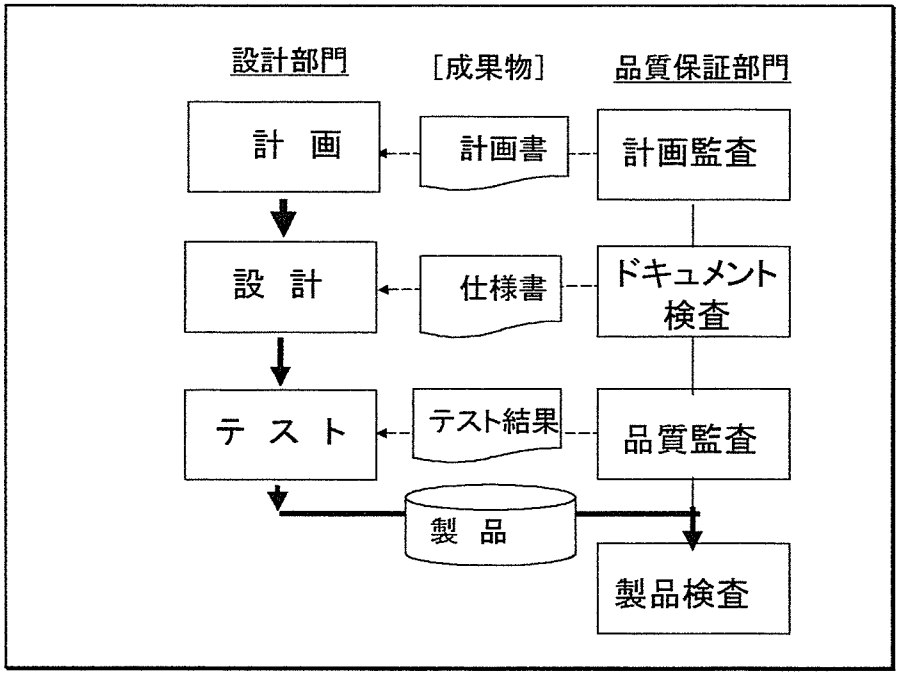


図２．１　品質保証部門の工程別役割

図２．２は，テスト工程において，ＰＤＣＡのサイクルを回す事例を示したものである．

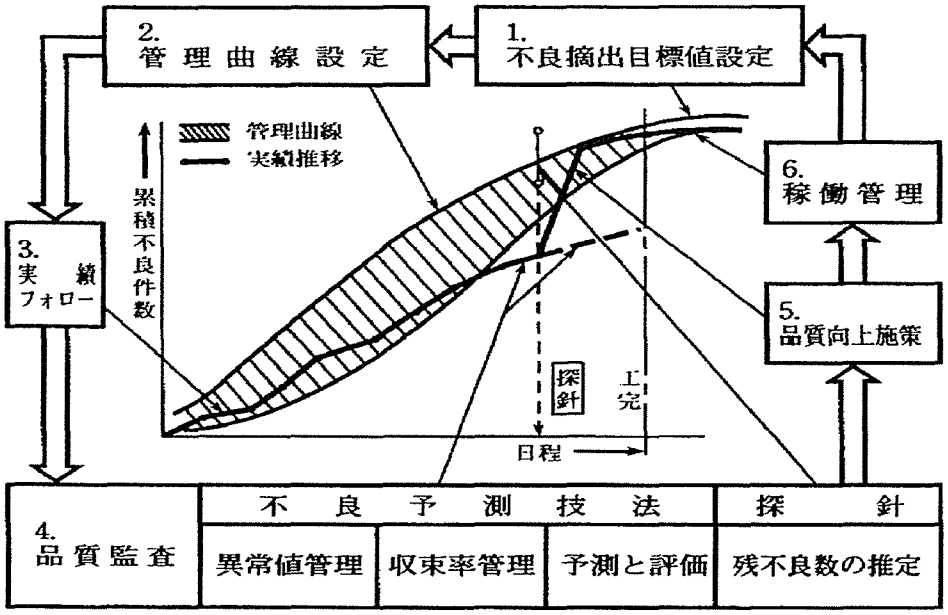


図２．２　テスト工程における品質目標値管理 [1]

（２）ソフトウェア工場[1],[4]

これは上記（１）で述べた開発方式を体系化した概念である．ハードウェアの品質管理の方法を，ソフトウェアにも適用しようというアプローチである．ソフトウェアの生産工程や品質管理のレベルを他の製品分野のエンジニアリングや製造工程のレベルにまで引き上げることをねらいとしてエンジニアと管理者を集め，これをソフトウェア工場と呼び，責任を明確にした．ハードウェアと同様に自動化を行い，検査と開発を分離し，管理の方法を定め，管理のための情報システムを整備し，部品化・標準化を行った．このソフトウェア工場については，米国マサチューセッツ工科大学のクスマノ(M.A .Cusumano)等[4]が詳細に調査している．

クスマノは．３種類の製品パターンと対応する開発方式を述べている．ハイエンド製品とは高度な軍事用システム等であり，クラフト方式である．一方，ローエンド製品とは大衆市場のニーズに合うＰＣ用ソフトウェア製品等である．これらも優秀なアーキテクトを含むアプリケーション指向プロジェクト方式が適している．日本のソフトウェア工場方式が適する分野として，上記両者の中間であるミドルエンド製品をあげている．

（３）あゆみ活動[8]

ソフトウェア開発は，通常自社内の要員だけで開発することは少なく，関連企業や外注先企業との協力で行われる．品質管理はこれら複数企業間の共同作業であり，どこか一つの組織が弱体でも問題が発生する．開発に参画している会社全体の品質管理，特にプロセスの管理を促進するための活動の代表例が「あゆみ」活動である．各社の管理状況，品質尺度を定義し，相対順位，各社の改善状況を調べ，フィードバックすることにより，さらなる改善を促進する方法である．

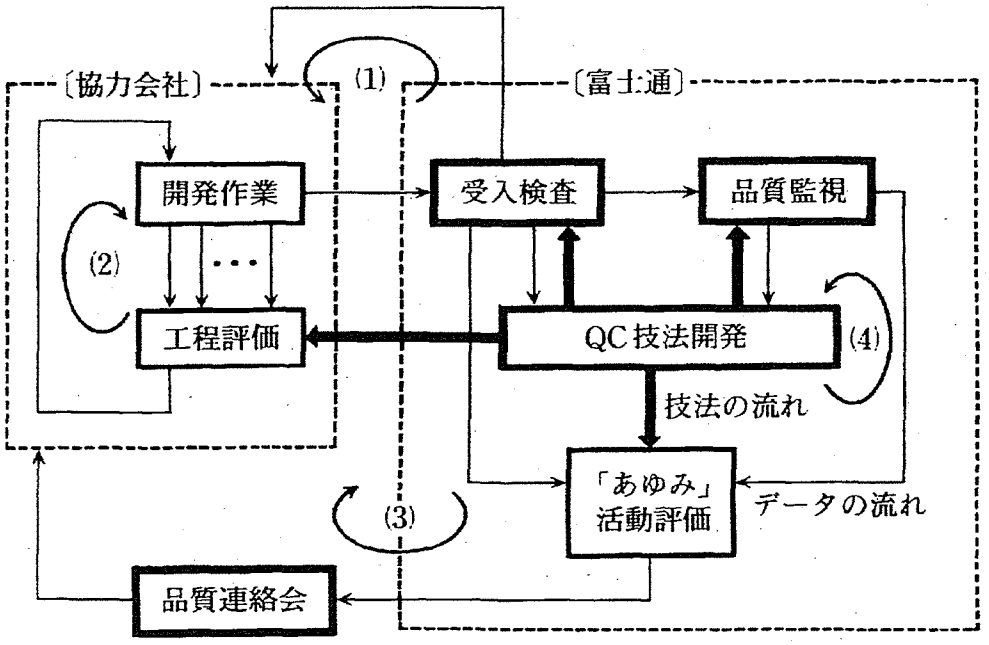


図２．３ あゆみ活動[8]

2.4 ソフトウェア・トラブルと発注者・供給者双方の留意点

ソフトウェア・トラブルといっても，分野によって，その様相はかなり異なるが，ここでは企業システムを主体とした「動かないコンピュータ」事例を分析した結果を基に，現場のソフトウェア信頼性の問題は何かについて検討する[1]．表２．１には，システム構築で稼動にこぎつけなかった過去１００件の事例に関する原因を示している．項番１と項番４の原因は，不良が除去できないという「狭義の品質の問題」である．項番２は，業務知識・理解不足という，いわゆる「要求分析・定義に関する問題」である．項番３は，システム計画が無理あるいは仕様不備という，「プロジェクト計画管理や上流設計の問題」である．これらで全要因の半数を占めている．

表２．２には，発注者・供給者双方の留意点を整理してある．業務システムの場合は，発注者側の影響が多いのが特徴であるが，ここでは，供給者（開発者）側の主要な問題点と留意点のみを述べる．

表２．１　「動かないコンピュータ」の事例[1]

項番	区分	原　　因	%
1	開発側	ソフト不良多発で完成できず	１５
2	開発側	開発者の業務知識・理解不足	１３
3	両者	システム計画が無理，仕様不備	１１
4	発注側	システムの検収が不十分	１１
5	開発側	運用するための納入者の指導不十分	9
6	発注側	ユーザの専任要員体制不備	8
7	両者	パッケージソフトが業務に不適合	8
8	両者	責任分担や納期の契約が不明確	7
9	—	その他	１８

表２．２　ソフトウェア開発における発注者・供給者方法の留意点[1]

項番	留　意　点	発注者	供給者
1	無理のない情報化システム計画立案(範囲, 納期, 規模, 体制等)	◎	
2	しっかりした要求分析・定義と文書化(仕様変更減少)	◎	○
3	業務に精通した開発者の選定	◎	○
4	妥当な見積り（特に規模見積りの精度向上）		◎
5	明確な契約の締結（発注業務範囲，分担，納期）	◎	◎
6	適切なプロジェクト推進体制(含む発注者側責任者の明確化)	○	◎
7	適切なプロジェクト管理	○	◎
8	品質管理体制の整備		◎
9	正確な設計による仕様の作成		◎
10	しっかりした受入れテスト・検収	◎	

［注］◎：特に留意すべき点，○：留意すべき点

(1) 品質管理体制の不備 ——> 品質管理体制の整備
(2) 見積り誤り ——> 妥当な見積り
(3) 不十分なプロジェクト管理 ——> 適切なプロジェクト管理
(4) 仕様変更の多発 ——> 構成・変更管理
(5) 不適切な仕様 ——> ユーザニーズを反映した

		仕様の確定
(6)あいまいな契約	——>	明確な契約の締結

上記の(1)から(4)については，次節の2.5.1で述べる．

2.5 ソフトウェア品質向上施策[1]

2.5.1 開発時の品質向上施策

ソフトウェアの開発では，コーディング量の数%にのぼる不良を作り込む一方，これら不良の除去と確認のために全工程の３０～５０%前後の工数をテスト（不良修正を含む）に費やしているのが現状である．これは相当な手戻り作業であり，効率のみならず品質の面からも隘路となっている．

（1）不良低減の方針

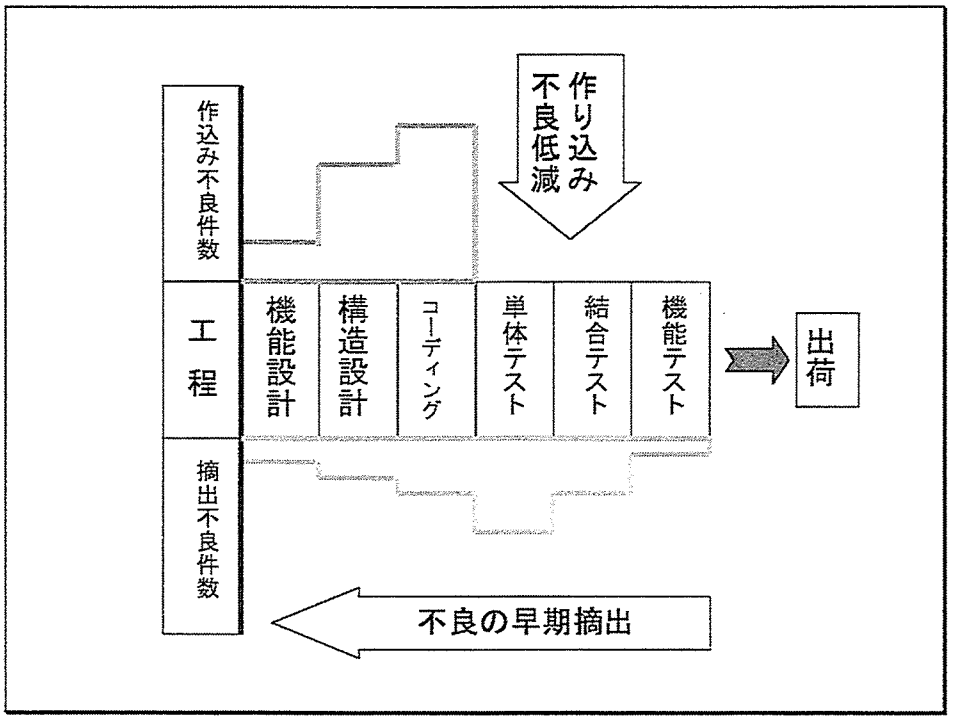


図２．４　不良低減のための方針図

ソフトウェアの生産工程における不良低減のための方針を図２．４に示す．現在のソフトウェア工学では，設計段階で不良の入り込みをゼロにするのは困難であるが，高品質を達成するためには次の２点をどれだけ徹底的に遂行するかが鍵となる．

●設計工程で不良の入り込みをいかに低減させるか
●不良をいかに早い段階で漏れなく摘出するか

これらの方針は「品質はプロセスで作り込め」という日本の誇る品質管理の基本原則である.

これを実現するためには,この30年間のソフトウェア工学の技法・知見やそれに基づく各種開発支援ツールを駆使することが必要であるが,ここでは触れない.

（2）Ｖ＆Ｖ（検証と妥当性確認）

これはソフトウェア工学の基本原則の１つである,上記（１）不良低減の方針が管理的立場からの方針とすれば,Ｖ＆Ｖ(Verification and Validation)は,それを達成するための技術的アプローチである.次に検証と妥当性確認の定義を示す.

検証(Verification)とは,開発プロセスへの入力に対してその出力が正しいかどうかを,各開発プロセスに対してチェックすることである.検証手段の代表的なものは,各種レビュー（デザイン/コード・レビュー,ウォークスルー,インスペクション等）と各種テストである.

一方,妥当性確認(Validation)とは,ソフトウェア開発工程の最後に,ソフトウェアが発注者の要求事項に従っているかどうかを確認するためにソフトウェアを評価することである.この手段は,使用者の運用に耐えることを確認するためのテスト（システムテストや受入れテスト等）である.

この２つを組み合わせることにより,ユーザ要求を正しく反映したソフトウェアの実現が可能になる.

上記の定義を図２．５で説明する.

まず検証であるが,機能設計工程を例にとると前段の要求定義工程の出力である要求定義仕様書がこの工程の入力になり,機能設計工程により,この要求定義仕様書がその出力である機能仕様書に正しく変換され,またその内容に矛盾がないかどうかをチェックすることである.

次に妥当性確認であるが,ソフトウェア開発工程の最終工程である受入れテスト工程で,最終ソフトウェア製品が要求定義プロセスの出力である要求定義仕様書の内容を満足しているかどうかを評価することである.

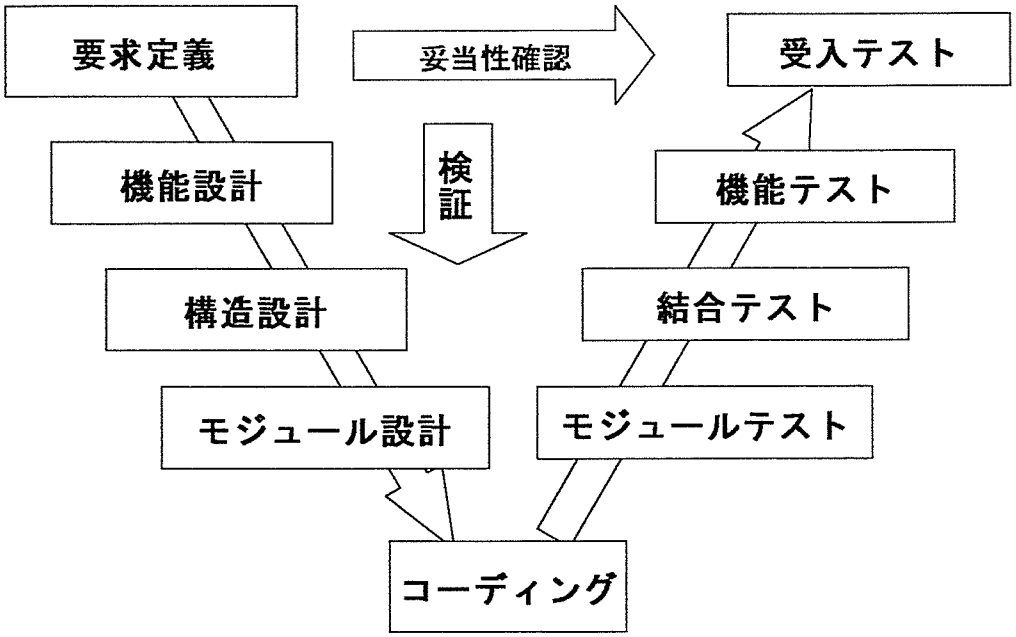


図２．５　Ｖ字モデルによるＶ＆Ｖの説明

（３）見積り技術の向上

大失敗プロジェクトの最大の原因は，見積りミスである．見積りの手順は，まず概略仕様から開発規模（通常はコーディング行数）を算出し，次に開発規模から開発工数を求める．ここで大きな問題点が２つある．第１は，一般に顧客の要求自体が曖昧なことが多く，仕様を固めるためには業務知識が必要になるということである．業務知識と顧客要求の適切な絞り込み技術を持つＳＥ（システムエンジニア）やソフトウェア技術者の養成が必要である．第２は概略仕様から開発規模を見積る技術である．元来規模見積もりは，高度な技術と経験が要求される分野である．一方，従来，規模の単位として使用されてきたステップ数（コード行数ＬＯＣ）が適用できないケース（部品やパッケージ使用等）が増大しつつある．また，最近はプログラムの量ではなく，ソフトウェアの価値を表す機能の量で評価すべきであるという考え方が有力になりつつある．これを定量化するファンクションポイント（機能量）法という手法が米国で開発され，これにより上記の問題も解決されるため徐々に日本でも適用されつつある．

（４）　プロジェクト管理

上記のような信頼性保証施策を遂行しても，納期やコストを守れなければ，プロジェクトは失敗する．一方，ソフトウェアが眼に見えないということは，その開発プロセス自体も見えないということであり，可視化に基づく管理が必要である．さらに，中大規模の開発になると数十人から数百人の開発要員が必要になる．プロジェクト管理が必須の所以である．

一方，近年対象分野を限定せず汎用的にプロジェクト管理技術を標準化する動

きが注目されている．米国 PMI(Project Management Institute) が作成したプロジェクト管理の知識体系 PMBOK(Project Management Body of Knowledge)[9]は，プロジェクト管理の内容を９つの分野に整理体系化した．さらに，ISO 9000の一環として「ISO/JIS Q 10006－品質マネジメントプロジェクトマネジメントにおける品質の指針」[10]が制定された．この枠組みも PMBOK とほぼ同様である．これらは，ソフトウェアに特化したものではないが，グローバルスタンダードとして今後普及すると予測される．

（５） 構成・変更管理

ソフトウェア開発においては，特に構成・変更管理が重要である．なぜかというと，どんなに要求定義をしっかりやっても，開発の途中で仕様変更が発生するのはソフトウェア開発の宿命であり，従って，変更管理をいかに上手に実施するかが要点になるのである．さらに，最近情報システムの主流になっているクライアント・サーバシステムでは，構成するハードウェア，ソフトウェアはいずれもオープン製品が主体であり，拠点ごとに随時各製品のバージョンアップが行われるため製品間のインタフェース不良がしばしば発生する．このため構成管理の導入が切実な課題になっている．

（６）部品化と再利用

これは，ソフトウェアが出現して以来，生産性向上のみならず品質向上のための変わらぬ課題であった．最近のオブジェクト指向技術の普及により，ようやく具体的な成果が出てきつつある．これをさらに推し進めたのが，ソフトウェアを開発せず，市販パッケージ・ソフトウェアを購入しようということである．しかし，これにも，失敗事例は多い．膨大なパッケージ・ソフトウェアの仕様を十分に調査し，業務が実現できるかどうか評価すること，また，実現のやり方については，業務をパッケージ・ソフトウェアの仕様に合わせることに顧客が同意していることが成否の鍵である．

2.5.2 開発プロセスの改善

（１）品質管理の進め方

管理のサイクル PDCA（Plan－Do－Check－Action）を回し，不良等の問題点の根本原因を追求し，これをフィードバックしてプロセスの改善を実現するというのが，日本企業のお家芸である．これはソフトウェア開発にも適用され，成果をあげてきたが，ソフトウェアの場合は，不良を修正しただけでは，根本対策とはいえず，従って再発防止策にもならない．これについては，次項で説明する．

（２）ソフトウェア不良の概念と定義

以下の３つの概念の違いを意識し，用語を使い分けて，データの収集・解析を行うことが肝要である．

●故障(failure)
[説明] ソフトウェアが期待通りに動作せず正しく機能しないこと．不具合の現象.
[使用目的] 顧客の影響把握
[例] システムダウン，結果不正
●フォールト(fault)
[説明] 故障を引き起こすプログラム内の誤り
[使用目的] 技術的要因の究明
[例] 排他不良，コーディング不良
●エラー(error)
[説明] フォールトを作りこむ原因となった知的活動(動機的要因)
[使用目的] 再発(類似不良)防止策に必要
[例] 命令語の理解不足，インタフェースの確認不足

故障やフォールトについては，ハードウェアの場合と概ね同様であるが，エラーの概念が異なる．つまり，この部分がハードウェア製品のように物理法則によらず,人間の思考に依存する製品の違いである．再発防止策を検討するためには，上記の「エラー(error)」の分析が大切である．

2.5.3 ノウハウの伝承

ソフトウェア開発は，目に見えない思考プロセスが主体であるため，開発が成功するかどうかは特にノウハウの適切な伝承如何に大きく依存している．ノウハウの伝承の順序としては，まず自分の組織内からであろう．ノウハウ集の活用が典型的なものである．

一方，最近，欧米で注目されているのが「ベストプラクティス」[11]，[12]である．他社の成功（失敗）事例に学ぶという方法である．ここでは，米国国防総省(DoD)が第一線の識者を集めて作成した「ソフトウェア開発のベストプラクティスとワーストプラクティス」を紹介する．これらの内容は，特に違和感は感じられないし，当然のことがあげられている．言い換えると，プロジェクト管理・品質管理の要点は，日米や時代の差が少ないということであろう．

（1）米国 DoD 推奨のベストプラクティス[13]
・組織的なリスク管理の実施
・インタフェース仕様の確定
・正式なインスペクション（レビュー）の実施
・メトリクスに基づいたスケジューリングと管理
・詳細化したプロセスに対する２値ゲート（完了か未完了）による進捗管理

・プログラム全体の予実績管理の可視化
・品質目標の設定と欠陥の追跡
・構成管理の励行
・メンバーの士気の向上と知識・成功体験の説明

（2）ワーストプラクティス警告[12]
・日程に余裕がないからという理由で新技術を導入するな
・提案依頼書(RFP)の段階で実装技術を規定するな
・十分に評価されていない「銀の弾丸」（新技術）を安易に擁護するな
・機能を大幅に減らさないで，大幅な納期短縮は期待するな
・クリチカルパスに含まれる作業項目をプロジェクト管理の対象外にするな
・過去の実績以上の大きな改善成果の達成を安易に期待するな
・複雑な仕様をすべてソフトウェアに押し付けるな
・ハードウェア・ソフトウェアの機能分割は，ソフトウェアの専門的技術なしに実施するな
・フォーマルレビューの参加者は５人程度までが望ましい．それを超えると超過人数に反比例してレビュー効果が低下する．

2.5.4 その他の品質保証施策

以上述べてきた品質向上施策以外にもISO 9000[14]に述べられているように，品質保証の立場から以下のような諸施策が必要である．

・組織の理念・経営方針
　顧客満足度（ＣＳ），信頼性重視の理念が明確に打ち出されており，これが経営方針で展開されていること．
・品質保証を実現できる組織体制
　欧米を含めてソフトウェアハウスでは，製造業のような独立した品質保証部門による検査が行われていない場合が多いが，高信頼性が要求される場合は，開発部門から独立した組織による検査が必要である．
・標準開発手順の採用
　各開発プロセスの定義（プロセスの入出力と処理）や作業詳細化(WBS)手順を組織として明確にすることが必要である．なお，この国際規格である「ソフトウェア・ライフサイクル・プロセス（ＳＬＣＰ）規格」を２．６で紹介する．
・生産技術・ツールと開発環境・設備
　ソフトウェア工学に基づく各種技術とその実現手段等は本論文ではふれていないが，当然重要である．
・躾と教育・訓練
ソフトウェア開発では，人への依存度が高いだけでなく，個人差が数倍に達す

ることもよく知られた事実である．凡ミス防止のための躾作法から使用言語・設計技法やプロジェクト管理の教育・訓練に至るまで，他の分野に比べても，人材の教育・育成は極めて重要である．

2.6 ソフトウェア開発管理の標準化動向[15]

2.6.1 ソフトウェア・ライフサイクル・プロセス(ＳＬＣＰ)規格 [16]

ソフトウェアが各種分野に浸透すると共に，ソフトウェアを開発・管理するための種々の標準，手続き，方式，ツールおよび環境が急速に多様化してきている．このため，ソフトウェアの開発管理が難しくなってきている．

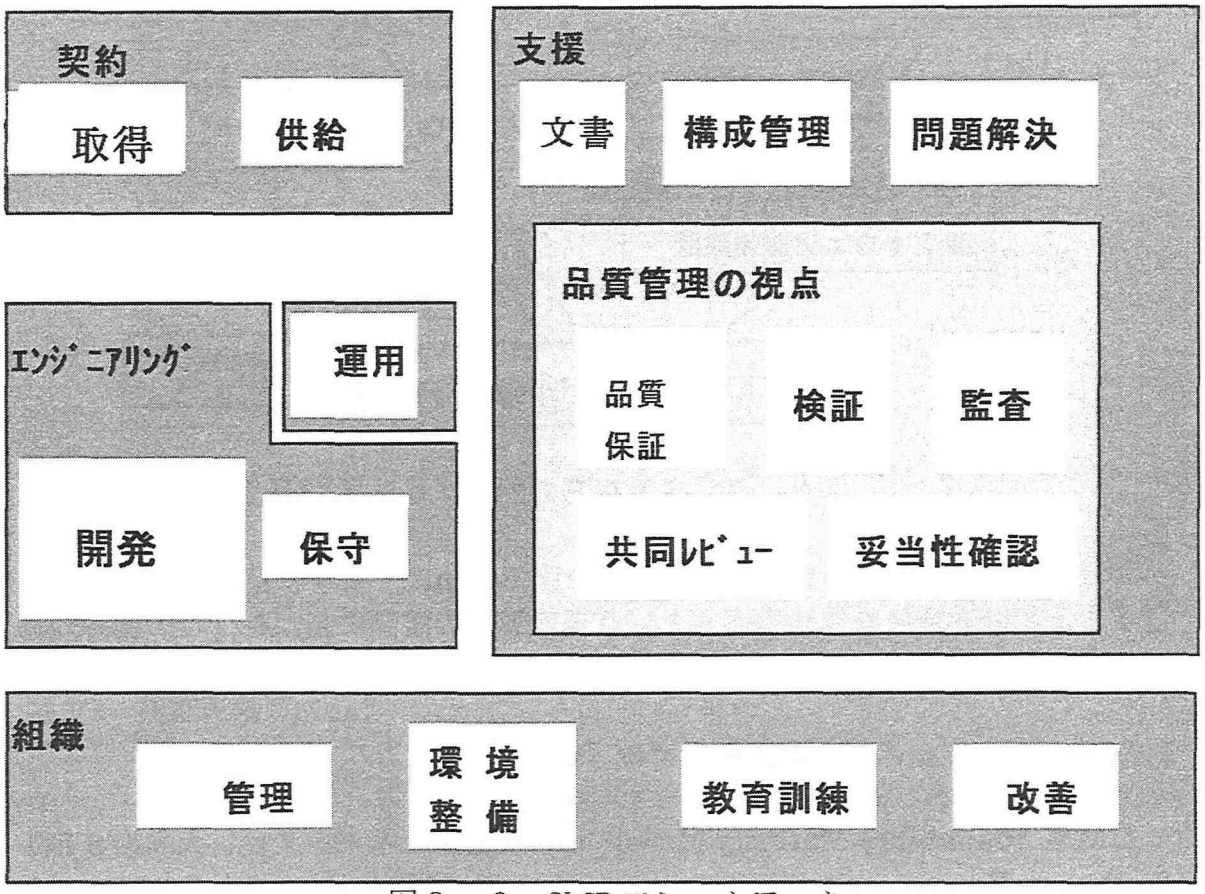


図２．６　SLCP フレームワーク

複数のソフトウェア製品の統合，企業間，国際調達プロジェクトなどの場合，特にその困難さが顕著である．そこで，ソフトウェアの開発・管理に際して，関係者が“同じ言葉を話す”ことができるように，共通の枠組みをもつことが必要となる．この規格は，こうした共通の枠組みを提供する．図２．６にこの枠組みの

概要を示す.

この枠組みは，ソフトウェアの構想から廃棄に至るまでのソフトウェア・ライフサイクルを包含し，ソフトウェア製品およびサービスの取得から供給までのプロセスを含んでいる．さらに，プロセスの管理および改善についても規定している．

この枠組みの構成は，プロジェクトレベルの契約・開発・運用・保守に関する主ライフサイクルと，支援ライフサイクル，および組織に関するライフサイクルプロセスの３つに大別される．図２．６には１７のプロセスが示されている．このプロセスは７４のアクティビティに展開され，さらに２２４のタスクに詳細化される．

図２．７に，開発プロセスのアクティビティを示す.

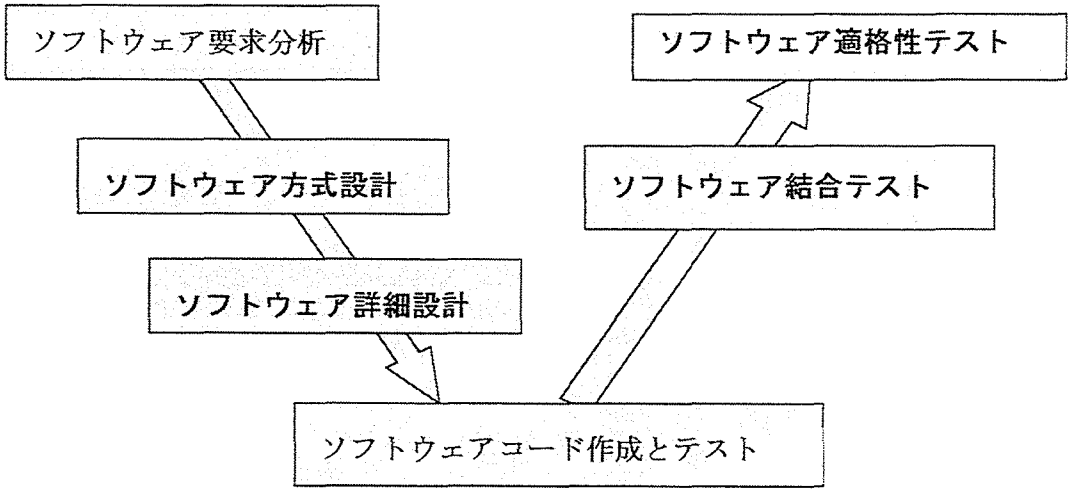


図２．７　ＳＬＣＰの開発プロセス

なお，我が国の情報産業の健全な育成を図るために，まず「共通の言葉」でソフトウェアの受発注を進めるべきという通産省への諮問に基づき，この国際規格をベースとして作成されたのが「共通フレーム９８ SLCP-JCF98」[17]である.

2.6.2　ソフトウェア能力成熟度モデル（CMM）[18]

ＣMM（Capability Maturity Model）は，米国カーネギーメロン大学のＳＥＩ(Software Engineering Institute)が１９９１年に開発したソフトウェアプロセス改善ガイドである．ISO 9000 は認証されるかどうかであるのに対し，ＣMMは組織の現状を評価し，次に何を改善すべきかを示してくれるため，欧米のソフトウェア開発組織で注目され，普及しつつある.

● ＣMMとは

ＣＭＭは，ソフトウェアの開発・保守のプロセスに対する管理能力の獲得を目指し，優れたソフトウェアの開発・保守と管理に向けた組織文化の発展を目指すソフトウェア組織のためのガイドである（この前提として，ソフトウェアの品質は開発・保守のプロセスの質に左右されるという認識がある）．その目的は，現状のソフトウェアプロセス成熟度を判定し，ソフトウェアプロセスとソフトウェアの品質を向上させるために最も重要な課題を識別することにより，ソフトウェアプロセスを改善する戦略を立てるためのガイドとなることである．

● ＣＭＭの５つの成熟度段階

ＣＭＭでは成熟度を５段階に分けて定義すると共に各段階で達成すべき項目（キープロセスエリア）を規定している（図２．８を参照）．

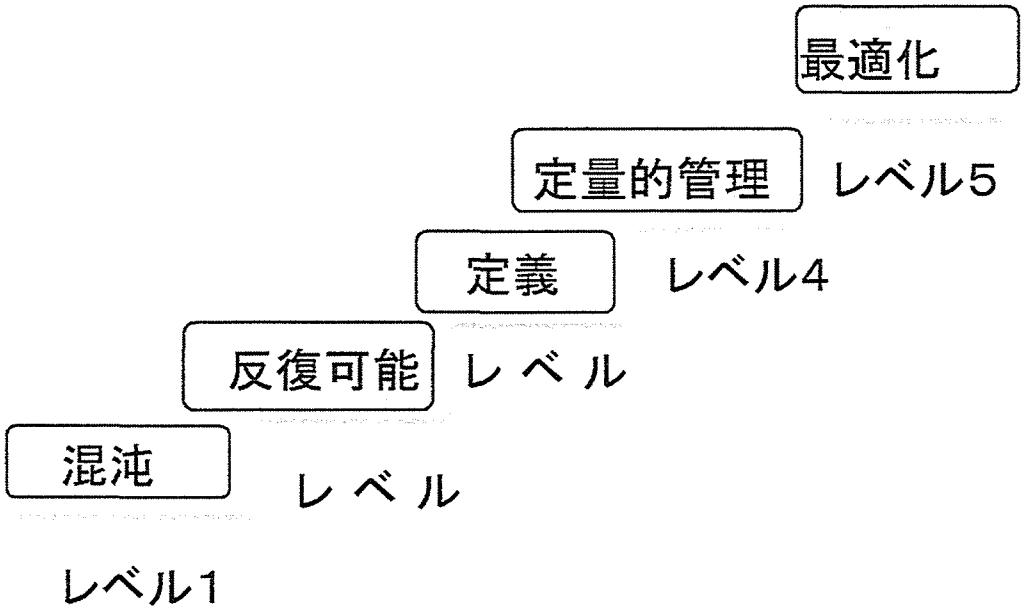


図２．８　ソフトウェアプロセス成熟度の５段階

（１）　初期段階(Initial)

ソフトウェアプロセスは場当たり的で，時には混沌としている．プロセスのほとんどは定義されておらず，プロジェクトの成功は特定個人の努力に依存している．

（２）　再現可能段階(Repeatable)

費用，スケジュール，および機能性をモニタするための基本的なプロジェクト管理プロセスが確立されている．プロセスが統制されているために，過去の類似したアプリケーションをもつプロジェクトの成功事例を再現することが可能となっている．

（3） 定義段階(Defined)
管理面と技術面の両方の作業プロセスが文書化され，かつ標準化されており，さらに組織の標準ソフトウェアプロセスに統合されている．全プロジェクトが，ソフトウェア開発・保守に関する組織の標準ソフトウェアプロセスの承認され，あつらえられた版を使用している．

（4） 管理段階(Managed)
ソフトウェアプロセスと成果物の品質について詳細な測定基準が存在している．ソフトウェアプロセスおよび成果物が定量的に把握され制御されている．

（5） 最適化段階(Optimizing)
プロセスからの，あるいは革新的アイデアや技術の導入例からの定量的フィードバックによって，プロセス改善が継続的に実現されている．

一方，ISO/IEC でも，これを参考にしてプロセス単位の成熟度の観点から規格化が進められている．

2.7 まとめ

最近の製品は，そのほとんどが組込みソフトウェアを内蔵していると言っても過言ではない．しかもその機能は増加の一途を辿っている．したがって，製品開発プロジェクトマネジャはもとより，ハードウェアの上級設計者も，ソフトウェア開発のあるべき姿と実態を知らないとプロジェクト管理をすることが困難であり，製品としての信頼性の要求を満足することも難しい．ソフトウェア技術者に対しても，グローバルスタンダード化が進みつつあり，2.6節で述べた標準化動向を理解し，積極的に取り組むことが必要になっている．

第３章　1990年代のソフトウェア品質マネジメント

3.1　はじめに[19]

日本のソフトウェア産業はコンピュータメーカ主導により，製造業で成功した品質管理を導入して，世界的にもトップクラスの高品質開発体制を確立した．しかし１９９０年代に入り，情報産業は，当時ネオダマと略されたネットワーク化，オープン化，ダウンサイジング化，マルチメディア化という大波のあおりを受けるのと同時並行的に，日本経済のバブル崩壊による不況というダブルパンチで打撃を受け，停滞した．さらに製造業としてのソフトウェア産業は，第２次産業から，インターネットビジネスに代表される情報サービスが主体の第３次産業へと変貌しつつある．また，パッケージソフトウェアを主体に，早く安い「そこそこ品質」ソフトウェアが市場を席巻しており，ソフトウェア品質管理の意義が問われている．２０世紀最後の２０００年秋，日本で開催された第２回世界ソフトウェア品質会議（２ＷＣＳＱ）での議論を基に，２０世紀日本のソフトウェア品質管理を振り返り，２１世紀を迎えるに当たっての課題を整理する．

3.2　これまでのソフトウェア品質管理の変遷

3.2.1　ソフトウェア品質管理の３段階の変遷

本格的にソフトウェア開発が行われるようになってきた１９７０年代から３０年間のソフトウェア品質管理の変遷を３段階に分けて述べる．これをまとめたのが表３．１である．

（１）3段階の時期の定義と特徴

●揺籃・確立期（～１９７０年代）

この時期はメインフレーム・コンピュータにより，定型業務をコンピュータでバッチ処理するのが主流であり，そのための業務ソフトウェアは使用するコンピュータに合せて個別受注して開発された．この時代のソフトウェア開発体制と品質管理を推進したのは，国産汎用コンピュータを開発販売したいわゆるメインフレームメーカであった．ソフトウェアの心臓部分であるＯＳ（オペレーティングシステム）や，社会的影響の大きい座席予約システム等の大規模オンラインシステムの開発には，ソフトウェア工学だけではなく，むしろ品質管理等の開発管理力をいかに高めるかが成否の鍵であった．ここで日本のメインフレームメーカは，いずれもコンピュータや通信機等のハードウェア機器を開発する製造会社であったため，当時日本の製造業発展の原動力であった品質管理をソフトウェア開発にも適用しようとしたのは当然のことであった．

表３．１　ソフトウェア品質管理の変遷[19]

年代	～1970年代	1980年代		1990～1995	1996～2000年
ソフトウェアの利用形態	メインフレームによる定型業務集中処理	オンラインシステムの普及		ダウンサイジング・オープン化	インターネット・サービス
情報産業の特徴		人材確保と生産性向上	88都銀第３次オンラインWS	クライアントサーバシステム ＰＣの普及	インターネット遍在化，電子商取引，ＡＳＰ，ディジタル家電，2000年問題
ソフトウェア品質管理	揺籃・確立期	発展・隆盛期		停滞・再構築期	
	メインフレーマ中心にソフトウェア開発管理体制（ソフトウェアファクトリ）整備 69年ソフトウェア工場（日立）	ソフトウェアハウスへの展開		ネオダマへの対応	ソフトウェア品質思想の多様化
		80年　ＳＷＱＣ活動（NEC）		ソフトウェア管理のグローバルスタンダード化 ・ＩＳＯ９０００－３（ソフトウェアへのガイド） ・ＳＬＣＰ(ｿﾌﾄｳｪｱﾗｲﾌｻｲｸﾙﾌﾟﾛｾｽ)規格 ・ＣＭＭ（ｿﾌﾄｳｪｱﾌﾟﾛｾｽ成熟度ﾓﾃﾞﾙ）	
日科技連活動		'80年日科技連ＳＰＣ研究委員会発足 ・ＳＰＣセミナー／ＳＰＣシンポジウム	ＳＰＣ研究会 '89ＳＰＣ海外調査団	ソフトウェア品質管理成果の普及推進	95第１回世界ソフトウェア品質会議（米国） 2000第2回世界ソフトウェア品質会議（日本）

年代	～1970年代	1980年代		1990～1995	1996～2000年
出版	79ソフトウェアエンジニアリング(菅野)		86ソフトウェア品質管理ｼﾘｰｽﾞ(SPC) '89富士通におけるソフトウェア品質保証の実際	90ソフトウェアの品質管理事例集(SPC) '90ソフトウェア品質管理ｶﾞｲﾄﾞﾌﾞｯｸ '90のソフトウェア総合的品質管理(NEC) '92実践ソフトウェア開発工学ｼﾘｰｽﾞ(SPC) '92富士通における「あゆみ」活動 '94 21世紀へのソフトウェア品質保証技術(SPC)	95ソフトウェア品質保証の考え方と実際(保田) '96ソフトウェアＩＳＯ9000(SPC) '99ソフトウェア開発体質改革論(金子)
品質管理	日本的品質管理	ＴＱＣ国際化		ＴＱＣ(管理)からＴＱＭ(経営)へ	
	51 デミング賞創設		87 ＩＳＯ９０００制定 '87 ＭＢ賞	ＩＳＯ９０００審査登録	ＴＱＭ宣言 日本経営品質賞
日本経済	業容拡大・輸出拡大	高度成長・経営の多角化		バブル崩壊	業務革新・経営のｽﾋﾟｰﾄﾞｱｯﾌﾟ

この方式はソフトウェア工場（ファクトリー）方式として後に欧米から注目を集めることになった．

●発展・隆盛期（１９８０年代）

この時期は日本経済の全盛期であり，都市銀行の第３次オンラインシステムに象徴される大規模なシステム開発が次々に行われ，メインフレームメーカだけではなく，多くのソフトウェアハウスが設立され，競ってプログラマを雇用した．このままではプログラマが不足するという「ソフトウェアクライシス(software crisis)」が真剣に議論された．一方，系列や外注先であるソフトウェアハウスの生産体制の整備が急務であるとして，ソフトウェア品質管理の技術移転が活発に行われるようになったのも，この時代である．この活動に大きく貢献した日本科学技術連盟のＳＰＣ（ソフトウェア生産管理）研究委員会が発足したのは１９８０年であり，研究委員会だけでなくＳＰＣセミナーやシンポジウムさらに研究会等を通じてソフトウェア品質管理の普及が精力的に行われた．１９８９年には第１次ＳＰＣ海外調査団が欧米を訪問し，訪問先では日本のソフトウェア工場方式とその高い実績が大きな関心を呼んだ．

●停滞・再構築期（１９９０年代）

この時期はバブルが崩壊し日本経済が失墜した時期であると同時に，情報産業は，「ネオダマ」（ネットワーク，オープン化，ダウンサイジング化，マルチメディア化の略）と言われる大変化に見舞われた．「ダウンサイジング化」とは，メインフレームという大型計算機から，サーバとワークステーション（ＷＳ）やパソコン（ＰＣ）を組み合わせたクライアントサーバシステム（ＣＳシステム）への移行である．さらに，従来各社のメインフレーム間でハードウェア・ソフトウェアの方式が異なり相互互換性がなかったのに対し，ハードウェアとソフトウェアの各構成要素間のインタフェースが公開され，それらの間の組み合わせを使用者が選択できるようになった．これが「オープン化」である．この時期は，ソフトウェア技術の変革期であった．オープン化やネットワーク化を支えるソフトウェア技術として，まずパソコンのＧＵＩ(グラフィカル・ユーザ・インタフェース)から「オブジェクト指向技術」が導入され，いろいろな局面に普及していった．オフィスでのＬＡＮ環境やソフトウェアの分散開発環境下で，グループウェアが脚光を浴びたのもこの時期である．高信頼性であるが高価格で製品選択や使い勝手の自由度が低く中央集権的構造のメインフレームコンピュータシステムは，急速にレガシー（過去の遺産）化し市場価値を失った．その後，さらに，インターネットの劇的な普及により，ネットワーク・コンピューティングと言われるように，ＷＳやＰＣも主役の座をネットワークに明け渡そうとしている．

（２）大変化のソフトウェア品質管理への影響

こうした大きな変化がソフトウェア品質管理に与えた影響は何か．詳細は後述するが，第１は，オープン化によりパッケージソフトウェアが急速に普及し，中身の分からない「ブラックボックス化」への対応を迫られたことである．

第２は，経営環境の変化により，「安く，早く」ソフトウェアシステムを構築す

ることが要求されるようになったことである．従来日本では，顧客独自の機能を高信頼性で実現することが要求されていたが，価格や納期も品質とトレードオフの関係にあるという分野が広がっている．品質思想の多様化である．これらに対応した品質管理の考え方とその方式を明確にしていく必要がある．

さらに，第３として，インターネットの普及により，情報産業の主体は，ソフトウェア開発から，ｅ－コマース，ｅ－ビジネスと言われるインターネットを介したサービスビジネスに移行しつつある．こうした新しいサービスの品質については２１世紀の課題である．

（３）グローバルスタンダード化の時代

一方，１９９０年代はグローバルスタンダード化の時代であった．その代表的な国際規格はＩＳＯ ９０００品質保証規格やＩＳＯ １４０００環境規格である．これらは，いずれも国際的な相互認証の仕組みをもつため，ビジネスへの影響力も大きい．デファクトスタンダードではプロジェクトマネジメントのＰＭＢＯＫがある．ソフトウェア品質管理の分野では，ＩＳＯ ９０００以外にも，ソフトウェア開発の枠組みを規定するソフトウェアライフサイクルプロセス（ＳＬＣＰ）規格や，規格化が進められている能力成熟度モデル／ソフトウェアプロセス評価（ＣＭＭ／ＳＰＡ）がある．

（４）組込みソフトウェアの発展

さらに，日本の製造業の復権を賭けて取り組んでいるのが，急速に規模の拡大と高機能化の必要に迫られている，組込みソフトウェア（embedded software）の開発分野である．この分野で先行していた企業の伝送部門では，１９８０年代後期から組込みソフトウェアのリソースが急膨張し，プロジェクト管理と品質管理が重要な課題になった．これと並行してソフトウェア生産技術分野では，オブジェクト指向の導入やソフトウェア部品化，さらに共通開発基盤の整備が進められた．情報家電は言うに及ばず，最先端のＩＴ戦略製品ではいずれもその差別化機能・性能の実現が組込みソフトウェアにかかっている．この領域は，１９８０年代に日本が成功した「Ｈｏｗ」の技術と管理を生かせるところである．その成否の鍵は，ソフトウェア開発人材の確保と開発体制の整備である．

3.2.2 ソフトウェア品質管理研究会テーマの推移

日本科学技術連盟のＳＰＣ（ソフトウェア生産管理）研究会が毎年の例会で設定してきたテーマは，表3.2に示す通りである．ＳＰＣ研究会が発足した１９８５年から５年ごとに毎年の例会テーマを並べてみると，５年ごとの特徴が見えてくる．それがキーワードとして選んだ，管理技術・開発技術・情報技術である．１９８０年代後半は，前節で述べたようにソフトウェア開発全盛期で大規模開発が目白押しであった．そこで，品質問題や，プログラマ不足に伴う外注管理，グループウェア等の分散開発，さらに「動かないコンピュータ」が問題になりプロ

ジェクト管理の課題がクローズアップされた．１９９０年代前半は「ネオダマ」に象徴されるコンピュータ技術の変革期であり，新しいソフトウェア工学をいかに現場に持ち込むか，また不況期を迎え，生産性をどう向上させるかが関心事であった．１９９０年代後半は，まずグローバルスタンダード化の第一弾としてソフトウェアでもＩＳＯ ９０００の認証が話題になった．その後は，インターネットに代表される「ＩＴ時代」への対応が主要な関心事になり，現在に至っている．

表 3.2　日本科学技術連盟ＳＰＣ研究会テーマの変遷[19]

年代	１９８５－１９８９	１９９０－１９９４	１９９５－１９９９
技術区分	管理技術	開発技術	情報技術
例会テーマ	・品質管理と品質保証 ・品質評価とその技法 ・品質解析と信頼性 ・プロジェクト管理 ・外注管理	・標準化および教育・訓練 ・ソフトウェアの生産管理 ・ＣＡＳＥ環境とツール ・ドメイン分析，要求分析，設計，製造，統合，検査の各技法 ・見積り技法・コストモデル	・品質保証システムの構築・ISO 9000 ・最近のネオダマ技術 ・インターネット ・ネットワーク ・協業（コラボレーション）

3.3 １９９０年代日本におけるソフトウェア開発の問題点の構造[20]

オープン化に伴うソフトウェア品質問題を検討することを目的に，日本科学技術連盟のＳＰＣ（ソフトウェア生産管理）研究委員会の筆者を含めた有志が，ＮＳＱＴ(ネオダマソフトウェア品質管理タスクフォース)研究会を８回開催した．そこで出された問題点を親和図で整理した結果，次のような原因の要点が抽出された[2]．

（１）原因の要点

a)オープン化を中心とした外部環境の変化が起き，ユーザの要求も変化した．

b)これらの変化に対して，ソフトウェア生産技術や関連する管理技術の対応が遅れた．

c)ソフトウェア開発が，従来のすべて自前開発から購入の方向に変わりつつあるが，その変化の影響に対する開発方法の対応が遅れた．

d)開発プロセスも従来の方法を踏襲しようとしており，変化への対応が遅

れた.

e)工程ごとの技術については，仕様検討や基本設計などの上流工程で大半の問題が起きており，コーディングやテスト等下流工程では目新しい問題は発生していない.

f)予め計画段階で問題を予測し手を打つのではなく，開発中に問題が発生してはじめて．管理者も経営者も，技術者も問題を認識する.

g)変化を認識する仕組みが欠落しているか，感度が悪いため，事前に変化に対応できていない.

h)これらは，技術面，組織，人材育成などの長期的視点での変化への対応としての戦略が欠落しているからであり，これが主要な問題の１つである.

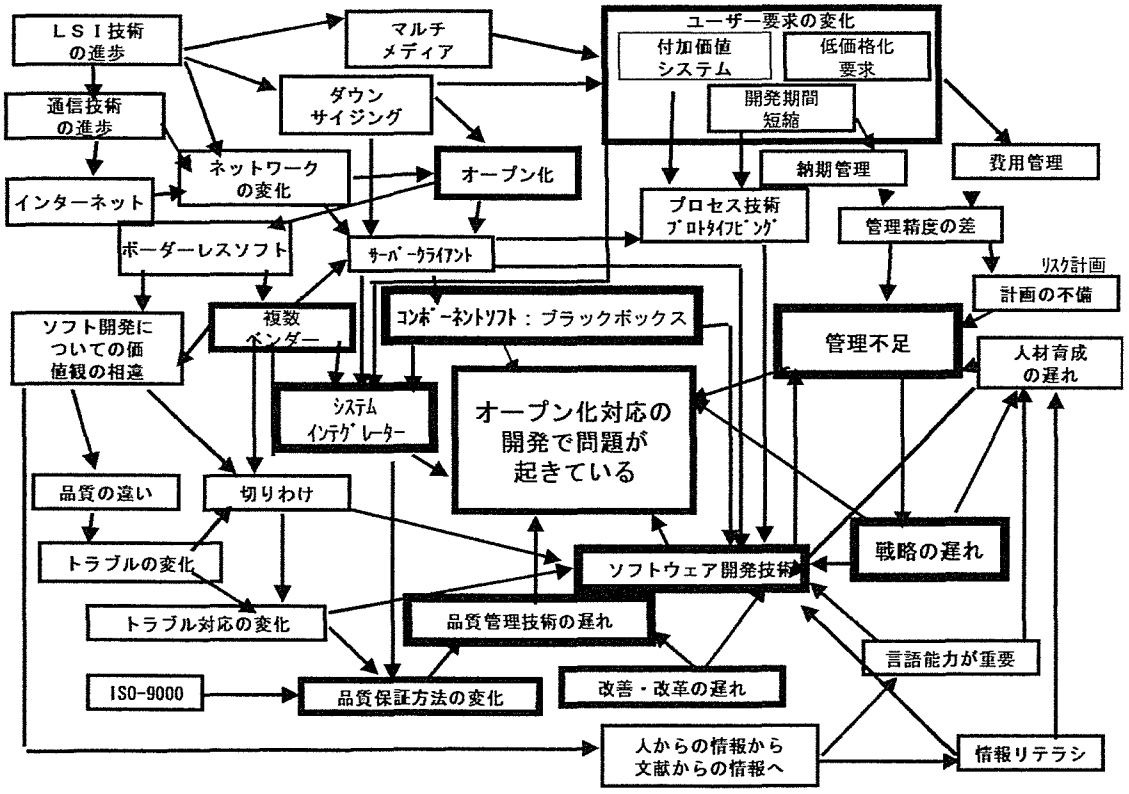


図3.1 日本のソフトウェア開発における課題の構造連関図

（２）問題点の構造解明

上記の様々な原因の構造を解明するために,新ＱＣ７つ道具のひとつである連関図[2]を作成した．これが図3.1である．これにより，日本的ソフトウェア開発における課題を抽出した．その結果，ソフトウェア開発技術，品質管理を中心とする管理技術，さらに戦略の遅れという問題点が浮かび上がってきた．この３点を次

の3.4節で検討する．

以上の変化と対応遅れから，オープン化時代のソフトウェア品質管理は課題が多い．これらは，オープン化に限らず変化への対応が遅れやすいことを意味しており，このままでは，これからも問題が再発する可能性が高い．

3.4　１９９０年代日本の課題と対策の検討[20]

3.4.1　技術の課題と対策案

３．３で検討したように，下記のような３つの大きな変化が生じ，そのために以下のような技術課題への対応が急務となった．

（１）ユーザが早く安くさらに付加価値を要求するようになった
これにより，付加価値を与えるための「顧客ニーズ分析・提案技術」，製品を早く安く提供するための「適切な開発プロセスの採用」，ユーザニーズや環境の変化に素早く追随できる「変更・拡張に強いアーキテクチャ技術」が必要である．
（２）オープン化により外部購入が拡大した
これにより，外部購入の際必要な「国際化対応技術」や，中身の分からない外部購入で必要な「外部購入品の的確な情報収集と評価技術」と「ブラックボックス技術」，「システム指向技術」が要求される．
（３）国際競争力を増強する必要がある
そのために，特許等の知的権利保護等に代表される「差別化技術」の推進や，従来日本のお家芸であった高信頼化のための「設計・検証技術」の復活が課題である．

以下に，上記技術課題のなかのキーワードの簡単な説明を加える．

（ａ）顧客ニーズ分析・提案技術
顧客からのニーズを的確に分析・把握する要求分析技術と，より積極的に顧客ニーズを先取りしてシステム提案できる技術が求められている．
（ｂ）適切な開発プロセスの採用
従来主流のウォーターフォール型プロセスに固執せず，開発条件に応じて最適な開発プロセスの採用が必要になる．外部購入品が主体の開発では，部品中心のプロセスの採用が必要になる．顧客ニーズに素早く対応して出荷するためには，段階的（インクリメンタル）な開発プロセスや進化型開発プロセスを採用すべきであろう．
（ｃ）変更・拡張に強いアーキテクチャ技術

仕様変更への対応および，構成要素であるソフトウェアとハードウェアの変更への対応が容易にできることが重要である．その技術課題は，変更容易性や拡張性である．この解決のためには，構成要素の部品そのものよりも，インタフェースを含むアーキテクチャの設計において変更容易性や拡張性が十分に考慮されていることと，構成部品を隠蔽する技術が重要である．

（ｄ）国際化対応技術

外部購入の拡大や国際分業開発の場合，英語力や明快な文章力，それにインターネットを駆使した情報収集・検索能力が必要である．国際分業開発の場合は，さらに，国際的に通用する仕様記述技術も必要になる．

（ｅ）外部購入購入品の的確な情報収集と評価技術

オープン化により，開発コストの低減や開発期間の短縮に対応するためには，各国からソフトウェア部品を外部購入せざるを得ない状況である．ハードウェア部品と同様に，組織的な外部購入品の的確な情報収集と評価が必須である．

（ｆ）ブラックボックス技術

ブラックボックスへの対応の基本は，事前評価である．実際の仕様を，詳細に知ること，評価できること，インタフェースを知っていること，切り分け技術があること，さらに使い方について種類，タイミングと頻度も知っていることが重要である．

（ｇ）システム指向技術

既にクライアントサーバシステムの構築でも必要であったが，パッケージソフトウェア等のブラックボックス製品を組み合わせたシステムでは，システム指向の技術が必要になる．たとえば，高信頼性を要求されるシステムでは，空間的，時間的冗長性を取り入れたシステム設計や検証法の採用を検討すべきである．

（ｈ）差別化技術

差別化の種類としては，第１に特許等の知的権利保護がある．さらに品質であるが，日本が得意なのは，「障害が少ないというような当たり前品質」である．価格・コストについては，ソフトウェアでは人件費の比率が高く，国内開発は海外との競争では不利になる．機能では，顧客ニーズ分析・提案技術が鍵である．性能は，アーキテクチャ設計がポイントである．さらに，タイミング（出荷時期）がある．

（ｉ）設計・検証技術

障害原因の１つに，仕様記述が自然言語であるための曖昧さがあげられる．仕様書の論理記述とそれに基づく設計段階の検証やソフトウェアレビュー手法であるソフトウェアインスペクションの実施が重要である．さらに，仕様記述の前提としてのモデリング技術がある．オブジェクト指向に基づくモデリングや正規表現，状態遷移記述等があげられる．

3.4.2 プロジェクト管理の課題と対策案[1]

管理の課題は色々あるが，ここではプロジェクト管理に絞って検討する．プロジェクト管理は，古くて新しい課題である．

日本のプロジェクト管理で伝統的に弱いのは，リスクマネジメントである．まず，事前にリスクを洗い出し吟味する必要がある．特に現実のプロジェクトでは，品質，コスト，納期を同時に満たすのが容易ではないことが多い．その場合，何を優先するのかは価値判断の問題であり，実行は意思決定の問題である．欧米では，“TRIAGE”という概念[21]で知られているもので，「乏しい資源を，そこから最大の効果を引き出すようなやり方で割り当てること」である．

2.4で説明したように，ソフトウェアで影響が大きい問題は，要求分析と見積り，それに変更管理である．情報システムの開発で，納期が来ても稼動できなかった原因を示す2.4の表2.1「動かないコンピュータ」の事例と，その対策を示す2.4の表2.2「ソフトウェア開発における発注者・供給者方法の留意点」[1]において既に述べた点である．

3.4.3 品質戦略の課題と対策案[20]

戦略の問題もいろいろあるが，品質戦略に絞って検討する．オープン化により，品質そのものの考え方，方針，戦略が変化している．この変化に気がつかなかったのがオープン化の問題点の１つであった．品質戦略には，ＺＤ型の「品質を重視」，ＣＳ型の「顧客の満足度を重視」，競争型の「市場制覇を重視」の３種類がある．

（１）ＺＤ型――品質を重視

最も品質の高いのがＺＤ(Zero Defect)型である．防衛関係や航空宇宙関係がその典型であり，日本市場では，公共性が極めて高いシステム，たとえば金融分野の基幹業務も含まれる．従来これらの産業では，かなりのコストをかけて欠陥ゼロの品質を追求してきた．その費用の原資は，政府予算や公共企業，独占型企業による費用負担であった．価値の基準は国家的ミッションの達成や国家・公共的組織のインフラ整備等である．この市場で生き残る企業の条件は，技術開発力と高品質であった．特定ニーズ向けであり顧客数や販売量が少ないため，価格は高くなり，開発経費は顧客負担である．開発企業では，バグゼロを目指してデザインレビューやソフトウェアクリーンルーム手法等によってプロセスを管理する．使用する技術も完成された技術を採用し，評価期間も長い．問題発生後ではなく，事前に未然防止を行うことが基本である．それでも問題が発生した場合は，真の原因を追究し，再発防止策を講じる．

（２）ＣＳ型――顧客の満足度を重視

顧客層を定義してその使用条件に合った範囲で品質保証を行い，経済性や開発期間などのバランスをとることが企業の方針である．開発企業では収益向上や開発の効率化のために商品のパッケージ化を行い，標準機能を使用する顧客には比較的安く販売する．一方，顧客別ニーズへの対応も必要であり，顧客別改造を容易にする設計が要求される．

（3）競争型――市場制覇を重視

この典型的ソフトウェアは，パソコン用パッケージソフトウェアである．顧客は限定されず，世界共通で億単位の顧客がいる場合もある．顧客指向型の開発はできず，仕様はメーカが決定し，販売期日も競争上決定され，ベータサイトテスト（製品を一般顧客に正式出荷する前に，限定顧客に試用して製品を評価してもらう制度）で，ある程度品質を確認し，販売する．すべての条件のテストは不可能であり，出荷後の品質は「そこそこ品質」“Good enough Quality”となる．

「そこそこ品質」[11]とは品質とスケジュール（納期）と機能性のバランス（トレードオフ）であり，その適切なバランスを決めるのはメーカではなく顧客である．

これらの品質戦略はメーカとして決まるわけではなく，また顧客ごとに決定されるわけでもない．市場も変化しており，以前と同じ戦略が通用するとは限らない．すなわち，市場の品質要求と企業の品質戦略および開発者の品質慣習というように，これら関係者の品質戦略が整合しないと問題が生じることを認識しておく必要がある．

例えば，パソコン用パッケージソフトウェアといっても，これだけパソコンが普及し，当初のパーソナルな使用形態を超えて情報システムや組込みソフトウェアにも採用されるようになると，そのＯＳのような基幹ソフトウェアは「そこそこ品質」では耐えられなくなる．現にこうしたパソコン用基幹ソフトウェアの開発では，設計者とほぼ同数のテスト担当者が投入され，その品質保証には膨大な工数が投入されていると言われる．

それでは翻って，日本のソフトウェア開発方式はどうあるべきであろうか．１９８０年代に注目を集めた「ソフトウェア工場方式」は，これらの品質戦略と整合がとれるのであろうか．この方式は，品質重視のＺＤ型製品には比較的うまく適用可能である．しかし，対象製品の大半を占める顧客満足度を重視するＣＳ型製品の場合は，以下のような開発方式の採用や工夫が必要になる．顧客ニーズの変化やタイムリな出荷時期の要求に対応するためには，ウォーターフォール型ではなく，段階的（インクリメンタル型の）開発方式を採用するとか，顧客との協調開発方式（ＪＡＤ：Joint Application Development）や迅速な開発を目的とするラピッドアプリケーション開発（ＲＡＤ： Rapid Application Development）を適宜採用することが要求される．そのためには，適切な開発方式を使い分けることを可能にする柔軟な開発組織や適切な管理の仕組み，教育訓練等が必要になろう．

3.5　２１世紀へ向けての課題

（1）１９９０年代の教訓

前節で，技術面，管理面（プロジェクト管理），戦略面（品質戦略）に関する課題と対策案を述べた．これらの問題点の多くは既になんらかの形で処理され，結果として解決済みとみなされているものが多いと思われる．しかしながら，それらの１つ１つは決して簡単なものではない．果たして真の意味で解決策が明確になっているのか，また再度類似の問題が発生した際には再発防止策がプロセスとして組込まれて有効に働くようになっているかどうか等を，組織の責任者はチェックしておく必要があろう．

（２）ソフトウェア開発の課題

ＩＴ革命の一翼を担う情報家電等を支える組込みソフトウェア系製品については，ソフトウェアの急速な規模の拡大と複雑さの増大という状況のなかで，「How」，すなわちどう作るか（信頼性や性能の実現等）の質が問われている．この分野は，元来日本の得意な領域ではあるが，この成否は日本の製造業の組込みソフトウェアに対する開発体制とソフトウェア要員の確保育成にかかっている．また早く安く良い品質をつくるノウハウを早急に獲得しないと諸外国からの追い上げに遭うことになる．一方，情報系ソフトウェア製品については，「What」すなわちニーズの把握と要求定義の技術の質が問われている．これらは従来のソフトウェア工学だけでは十分でなく，また類似仕様のソフトウェアを開発するのには適した「ソフトウェア工場方式」も見直しを迫られている．人間や社会組織を対象とする心理学や行動科学等の各種分野の知見が必要であろう．

（３）新しい技術動向や多様化する品質への対応

新しい技術としては，例えば，Linuxのような，オープンソース（ソフトウェアのソースコードが公開され，無償で使用でき，自由に手を加えることが可能）形態への対応がある．多様化するソフトウェア品質に対しては，その内容と重みへの的確な対応が要求される．例えばセキュリティ品質でいえば，ネット上での攻撃に対して「セキュリティホール(セキュリティ上の抜け道)」がないことなどが，新たな品質特性の要素としてクローズアップされるというようなことである．

（４）ＩＴ革命での新しいビジネススタイルへの対応

インターネットビジネスを牽引してきた情報産業では，その主体は今やソフト開発ではなく，ソリューションビジネスという名のサービス産業やインターネットビジネスに代表される新しいビジネスモデルに基づく新たに創出される産業形態である．この分野の「サービス」の「質（Quality)」とは何かについて，日本科学技術連盟ＳＰＣ研究委員会でも取り組み始めているが，まだその道筋が見えていないのが実状である．これこそは，２１世紀の初頭に取り組まなければいけな

い大きな課題である.

（5）技術の伝承

１９９０年代の急速な技術革新のなかで，ソフトウェアの管理者や中堅技術者の多くがその変化についていけず，自信を失って若手の指導が十分にできていないと言われている．これが１９８０年代までに蓄積されてきた開発技術・管理技術やノウハウの伝承を困難にしている．今後とも続くことが予想される変化のなかで，いかにして伝えるべき技術を伝承していくかは大切な課題である.

（6）社会体制・文化という視点からの考察

品質管理を含むマネジメントの領域は，社会体制や文化に立脚するという観点から，日本の経済戦略や日本社会の構造的特徴を踏まえた検討が必要である．また，組織論や企業文化という観点からの分析も必要である[22].

第４章　ソフトウェア品質マネジメントの枠組みの再構築と対策方針

4.1　１９９０年代のソフトウェア開発を取り巻く変化[19]

第３章での分析により，１９９０年代の日本経済の混乱と情報分野のオープン化時代を迎えて，日本のソフトウェア産業が変化に弱いことが明らかになった．混乱の原因となった変化は，以下の５点に集約される．

●日本固有の生産管理・生産技術や自社製品のみによる情報システム構築が困難（オープン化・ネットワーク化の進展に伴う多様な製品群の組み合わせによるシステム構築の実現要）
- ・コンピュータシステム構成の多様化と製品間インタフェースの複雑化
- ・開発体制・管理・技術のグローバル化への対応（従来方式の見直し）とグローバルスタンダードの導入
- ・自社製品以外の多様な製品群によるシステムの品質保証

●社会情勢の変化，技術革新のｽﾋﾟｰﾄﾞｱｯﾌﾟ，経営の不安定化の増大
- ・開発リスクの増大

●多様な価値観や国内外競争の激化
- ・適切なユーザニーズの把握と要求定義の必要性大

●低価格化・短納期化のもとでのプロジェクト開発の増加
- ・プロジェクト管理の失敗による納期遅延・品質不良・赤字製品の発生

●技術革新や人材の流動化，外注委託，外部購入等による，若年層への生産管理・生産技術の空洞化
- ・ソフトウェア生産管理・生産技術の伝承

4.2　ソフトウェア開発に対する課題

前節で述べた変化により混乱を招いたが，これらの変化に対して必要な技術や仕組み，管理面の課題は以下の通りである．

（１）開発体制・管理・技術のグローバル化への対応（従来方式の見直し）とグローバルスタンダードの導入の必要性

・「オープン化・ネットワーク化の進展により，日本固有の生産管理・生産技術や自社製品のみによる情報システム構築が困難」の原因から，当然必要となる．

（2）開発リスクの増大への対応
・従来日本ではあまり重視されなかったが，「社会情勢の変化，技術革新のスピードアップ，経営の不安定化の増大」から，切実な問題である．

（3）適切なユーザニーズの把握と要求定義の必要性
・これは従来から問題点として提起されていたが，「多様な価値観や国内外競争の激化」の理由から，緊急かつ重要な課題になっている．

（4）コンピュータシステム構成の多様化と製品間インタフェースの複雑化
・「日本固有の生産管理・生産技術や自社製品のみによる情報システム構築が困難（オープン化・ネットワーク化の進展に伴う多様な製品群の組み合わせによるシステム構築の実現の必要性)」の結果，１９９０年代半ばから発生している課題であり，設計・テストの両面からの対応が必要である．

（5）低価格化・短納期化のもとでの品質確保への対応
・「低価格化・短納期化によるプロジェクト開発の増加」に対して，開発技術面とプロジェクト管理の両面からの対応が要求される．

（6）プロジェクト管理の失敗による納期遅延・品質不良・赤字製品の発生への対応
・「低価格化・短納期化によるプロジェクト開発の増加」により，管理面の強力な対応が必要である．

（7）自社製品以外の多様な製品群によるシステムの品質保証
・「日本固有の生産管理・生産技術や自社製品のみによる情報システム構築が困難（オープン化・ネットワーク化の進展に伴う多様な製品群の組み合わせによるシステム構築の実現の必要性)」に対する品質保証の観点からの対策が必要である．

（8）ソフトウェア生産管理・生産技術の伝承
・「技術革新や人材の流動化，外注委託，外部購入等による，若年層の生産管理・生産技術の空洞化」に対し，OJT だけでなく，組織的・体系的教育・訓練による伝承が急務である．

以下では，本論文のテーマである広義の品質マネジメントに絞って考察する．

4.3　１９９０年代ソフトウェア品質マネジメントの課題に対する対策方針

4.2 で述べた「20 世紀末日本のソフトウェア品質マネジメントの課題」を解決するための本論文で議論した対策方針を，図４．１に示す．

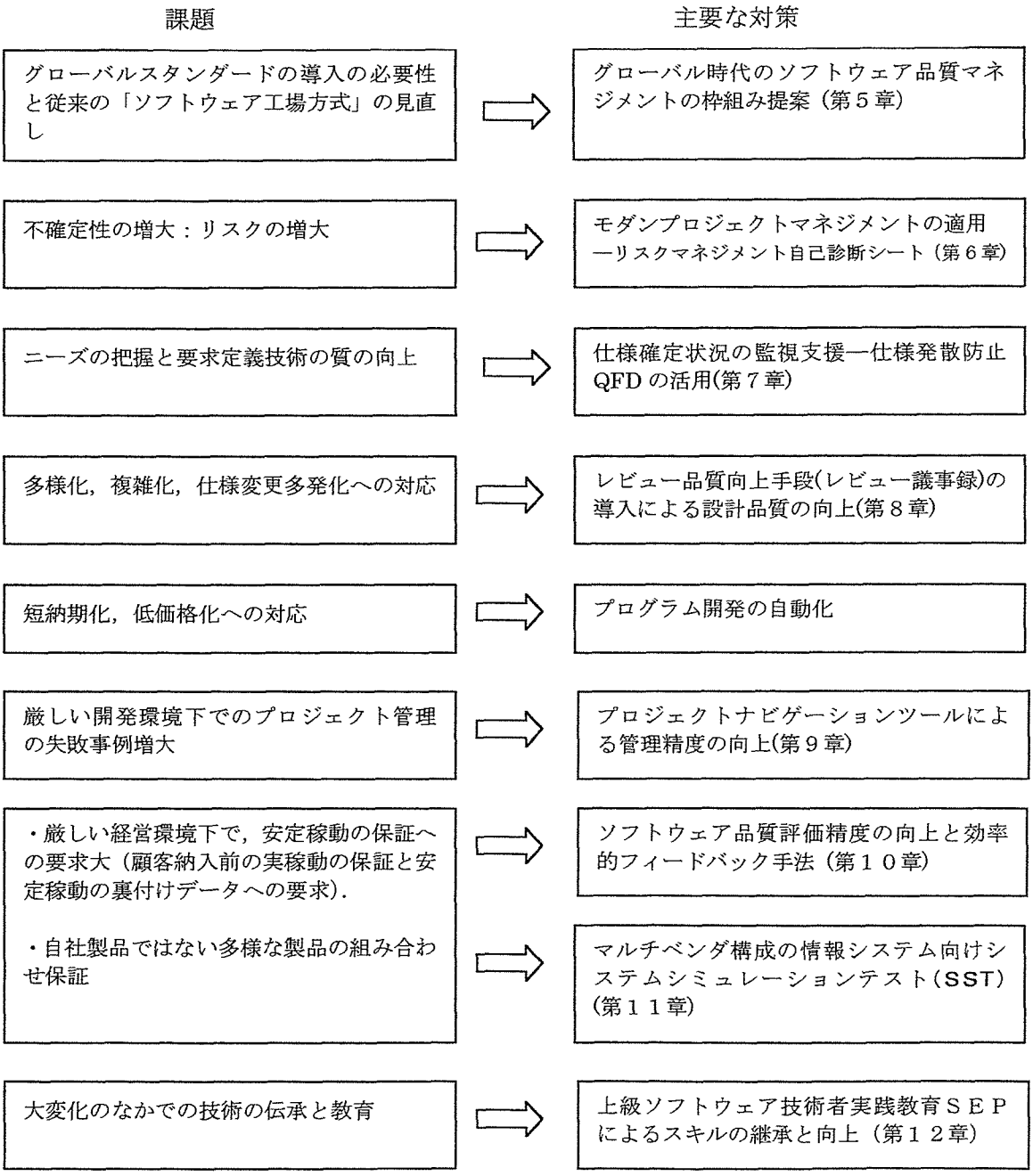


図４．１　20 世紀末ソフトウェア品質保証の課題に対する対策方針

上記の図４．１で示した主要な対策を第５章以下の各章で詳細に述べるが，その構成と概要を以下に述べる．

第５章は，グローバルスタンダードと整合する品質マネジメントシステムを示し，さらにこれを実現するための組織（プロジェクトオフィス機能を従来組織に適合するようにして導入）と仕組みを提案する．

第６章と第７章は，従来から日本のソフトウェア開発の弱点と言われてきた上流工程に対する対策である．

第６章は，従来のＱＣＤ（品質，コスト，納期）中心であったプロジェクトマネジメントに，６つの知識領域を加えたモダンプロジェクトマネジメントを採用することで，特に上流工程のリスク管理を強化する施策（リスクマネジメント事前自己診断シート）を中心に述べる．

第７章は，仕様の早期明確化と仕様変更に対する施策である．これはＱＦＤ（品質機能展開）という品質管理手法を応用した「仕様発散防止ＱＦＤ」を考案して，仕様確定状況の監視支援を行うというものである．

第８章は，レビュー品質向上手段を開発・導入することにより設計品質の向上を図るという施策である．各レビュー時に作成する議事録をデータベース化し，検討項目ごとに編集して仕様決定の経過を分かりやすく示すとか，標準レビューテーマと実際の検討内容を比較して検討不十分な項目をチェックできるようにするものである．

第９章は，プロジェクト管理の精度向上を図るために，「プロジェクトナビゲーションツール」を開発し，適用するという施策である．このツールは，ＷＢＳの標準化，成果物の一元管理と再利用，およびＷｅｂ／グループウェア技術の利用による分散開発の高効率化を狙いとしている．例えば，プロジェクト管理者にとっては，開発者からの特別な報告がなくても状況把握ができるとか，成果物の現物確認が容易である等の効果がある．

第 10 章は，テスト段階の品質評価精度の向上と効率的なフィードバックを可能にする手法を提供する．具体的には，複数の指標の相関評価とそのパターン化や，評価要素の目標値達成度による評価点法の導入により，比較的簡単に品質評価ができる．

第 11 章は，マルチベンダ構成の情報システム向けシステムシミュレーションテスト（ＳＳＴ）の導入である．ＳＳＴとは，それぞれ検査合格した製品を自社内のセンタに持ち込み，出荷前に顧客システムを擬似した環境下で行うテストである．自社製品主体のＳＳＴは 20 年以上から実施しており，効果をあげてきたが，近年マルチベンダ構成や多岐にわたるネットワーク網を利用した情報システムが増加し，従来型ＳＳＴでは実現が困難になってきた．そこで，他社製品やネットワーク網の投資と共にテストノウハウを蓄積し，マルチベンダ構成ＳＳＴを実現した．この具体的事例と成果を示す．

以上第 6 章から第 11 章は，いずれも開発プロセスの改善について述べた内容であるが，次の第 12 章は人材育成に関するものである．これは，ソフトウェア開発リーダ向けの開発技術と管理の両面を，講義とミニプロジェクトでのグルー

プ演習を通じて習得する長期研修である．これは 1990 年から開始し，改良を加えてきた．この研修の内容の詳細を述べると共に，評価を試みる．

第５章　グローバル時代のソフトウェア品質マネジメント

第４章で述べた対策方針を具体化したフレームワークを以下に述べる．これは，以下の２つからなる．

・品質マネジメントシステムの視点からのもの
・組織の仕組みの観点からのもの

5.1 グローバルスタンダードと整合する品質マネジメントシステム

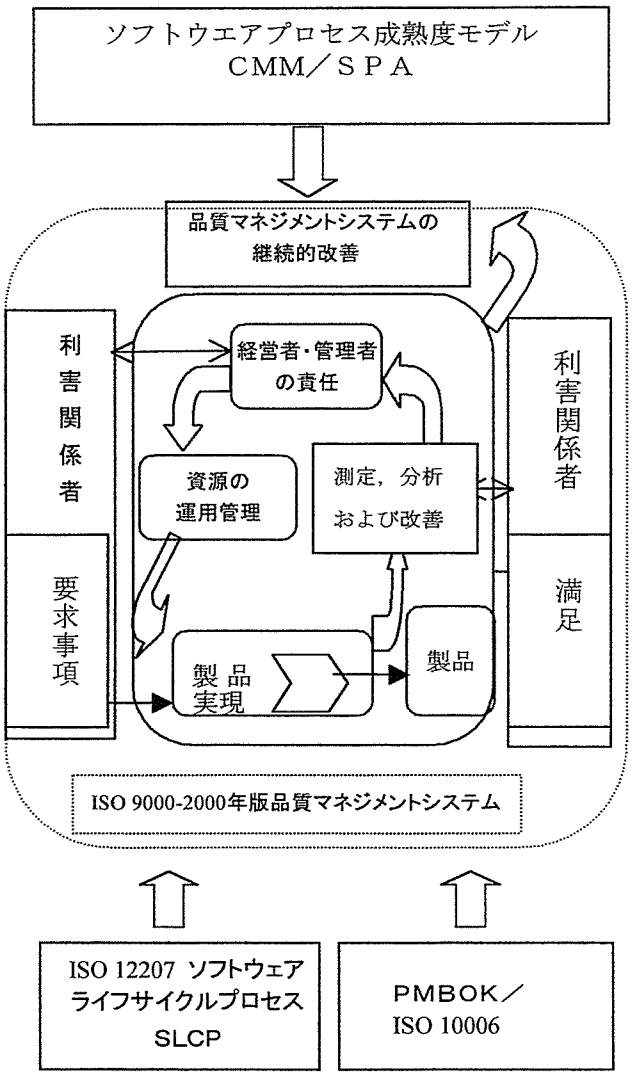


図 5.1 グローバルスタンダードと整合する品質マネジメントシステム

本節のタイトルである「グローバルスタンダードと整合する品質マネジメントシステム」を図示したのが図 5.1 である．図 5.1 を次に説明する．中核に据えたのが，２０００年改訂の ISO 9004 対応の「JIS Q 9004:2000 品質マネジメントシステムーパフォーマンス改善の指針」[23]である．製品開発プロセスのベースとしては，図の左下に示した「ISO/IEC 12207 ソフトウェアライフサイクルプロセス (SLCP)国際規格」[16]を採用する．さらに，プロジェクトマネジメントの視点については，図の右下に示したデファクトスタンダード「プロジェクトマネジメントの基礎知識体系（ＰＭＢＯＫ)」[9]あるいはほぼ同等の ISO 10006[10]を適用する．さらに「品質マネジメントシステムの継続的改善」のモデルとして，組織レベルのプロセス改善については，デファクトスタンダードといえる「ソフトウェアプロセスの能力成熟度モデル（ＣＭＭ)」[18]を適用し，プロセス単位の改善については，ISO/IEC JTC1 で規格化中の「ソフトウェアプロセスアセスメント（ＳＰＡ)」規格案を採用する．なお国際規格の適用にあたっては，規格が認める範囲内で当該組織に合うようにテーラリングあるいはカスタマイズすることができる．また，ＳＬＣＰ規格の場合には，各プロセスの名称等は当該組織が従来から使用しているものをあえて変える必要はなく，組織の標準開発手順とＳＬＣＰ規格の対応関係を明確にして比較表を作成することが現実的である．

5.2 「品質マネジメントシステム」を実現するための組織と仕組み

「品質マネジメントシステム」を実際の組織に展開する場合は，開発対象システムの性格を意識して最適な形態にする必要がある．ここで提示するのは，ある程度大規模で高信頼性が要求される特定顧客向けソフトウェアシステムの開発組織である．

これは，１９８０年代に世界的に評価された日本独自の製造業の開発形態をモデルにした「ソフトウェア工場方式」[1],[4]をベースとしたファクトリシステムと,プロジェクト方式の長所を生かしたより柔軟なハイブリッド方式である．以前のファクトリシステムと比べて経営層による強力かつ迅速な指導(特に経営面のリスクに対して)を可能にする仕組みであることが特徴である．

通常のプロジェクト方式では，プロジェクトマネジャーが品質・コスト・納期のすべての責任を持ち，管理する．本提案方式では，品質に関しては品質保証部門が独立した立場から活動を行い，最終的な製品検査では合否判定，従って出荷の可否を品質保証部門長が行う．これは，ファクトリシステムを踏襲したものである．通常のプロジェクト管理は，プロジェクト内で行うが，当該事業部門の経営に関わるようなリスクについては，一般的にはプロジェクトオフィス機能と呼ばれる，事業部門組織に属するプロジェクトマネジメントセンタで見積り，受注

段階から強力に監視する．この仕組みは以前の「ソフトウェア工場方式」にはなかったものである．これを示したのが，図 5.2 である．

これは，最近世界的に注目を集めている「エンタープライズ・プロジェクトマネジメント（EPM)」の一種であるとも言えるが，それは，企業の経営向上に効果をもたらす手法であるという点である．しかしながら，ベースとなるのが「ソフトウェア工場方式」であり，これは，通常のプロジェクト方式ではなく，むしろ機能別の伝統的な工場システムの枠内でのプロジェクト制度をベースとした組織における，プロジェクトオフィス機能の導入という形態である．

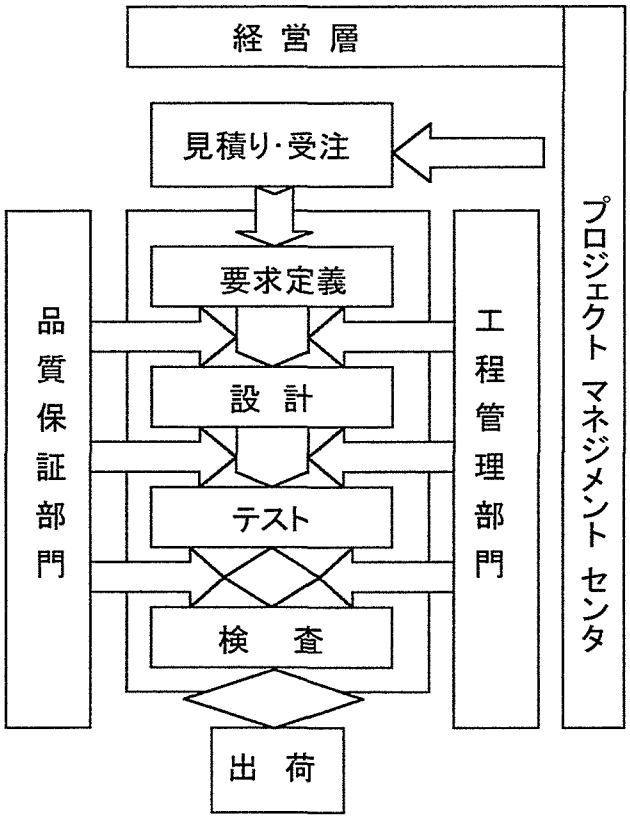


図 5.2 ファクトリ・プロジェクト・ハイブリッド方式

新しいフレームワークは上記の通りであるが，より具体的な手法としては，以下のような改良点がある．

[具体的適用事例]

プロジェクトマネジメントセンタ設置による事業組織でのリスク管理強化

従来から，ソフトウェア工場制度により，個々のソフトウェア開発プロジェクトに対して，品質保証部門による品質面からのチェックや工程管理部門による進捗管理は行われていた．しかし，昨今の状況では，プロジェクト発足以前の見積

りや受注契約段階でのリスク管理の強化が必要となってきた.

また，開発段階においても，大幅な仕様追加・変更や納期の変更要求が事業の根幹に影響を与えるケースが増加しつつある．そこで，一般的には「プロジェクトオフィス」と呼ばれる，事業組織レベルでのリスク管理を主体とした「プロジェクトマネジメントセンタ」を設置した．設置前後の効果比較をするには，時間が不足しているが，従来と比べて，経営レベルでの適切な判断が迅速に行われるようになりつつあると考える．

5.3 伝統的品質マネジメントシステムへの改良

5.1 と 5.2 では，グローバル時代に対応可能な品質マネジメントシステムの枠組みと組織の要点を述べたが，第４章で述べたとおり，ソフトウェアの各開発プロセスにおいても各種の施策が必要である．第６章以降でそれらの代表的施策を述べるが，それ以外にも，一般的に知られているが必ずしも普及していない効果的な方法論や手法がある．これらのいくつかについて，以下に，簡単に述べる．

（１）「開発スピードの向上」と「変化への対応」を柔軟に行う開発方法論の採用

従来の主流である，最初に全体仕様を決定しこれを詳細化していくというウォーターフォール型開発ではなく，例えば，まず核になる部分を開発し，それに段階的に追加していくという段階的（インクリメンタル型）開発の採用である．

（２）顧客満足度の重視（従来の「How」主体から「What」重視へ）

ユーザニーズの適切な製品機能への反映のために，たとえば 品質機能展開（ＱＦＤ）の採用がある．

（３）手戻りの少ない開発

手戻りの少ない開発を行うためには，設計段階の不良を早期に摘出することが必須であり，ソフトウェアインスペクション等の効果的なレビュー技法の導入がある．第８章で述べる設計レビュー議事録のデータベース化と検索ツールの活用もその１つである．

（４）フィードバック制御に加えてフィードフォワード制御の強化

従来出荷以前に適用していた品質予測を強化し，出荷後の品質予測を行い出荷可否の判定にも利用可能にする． 出荷以前の品質予測の強化手法と，出荷後の品質予測の具体的事例は第１０章で述べる．

以上のほかにも，第４章で述べた施策のいくつかについて，以降の章で詳細に説明する．

第6章　モダンプロジェクトマネジメントの適用

近年，ＩＴシステム開発プロジェクトは，多様化・複雑化し，短期開発が求められると共に，開発中の要件変更も増加している．ＩＴシステムは，その構築中，構築後を含めて，その全体像を直接目にすることができないという特徴をもつ．

このような状況の中で，日立製作所ではリスクマネジメント，スコープマネジメントを中心として，モダンプロジェクトマネジメントの適用を図っている．ＩＴシステム開発プロジェクトへのモダンプロジェクトマネジメント適用の具体例を示すと共に，今後の課題を考察する．

6.1 はじめに

最近のＩＴシステムの構成要素は，急速な発達が進むネットワークおよびパソコンなどを含めて，複数ベンダが納入するケースが増加している．

ビジネス面からの要件は，社会および企業の隅々にＩＴ化が及ぶと共に，国際化を始めとするビジネス環境の急激な変化，企業間の連携の進展などから多様化し，複雑になり，短期間で変化するようになってきている．そのため，数メガラインにも及ぶアプリケーションプログラムを短期間で開発することが求められ，開発中の要件の変更も増加している．

このような変化によって，ＩＴ開発プロジェクトの一部には，原価高，工程遅れ，品質不足などの問題を抱える，いわゆる混乱プロジェクトも散見される．ＩＴシステム開発プロジェクトでは，従来，「原価」，「工程」，「品質」に着目したプロジェクトマネジメントを行ってきたが，上記のような背景から，PMBOK[9]に代表されるモダンプロジェクトマネジメントの知識体系を，ＩＴ開発プロジェクトのプロジェクトマネジメントに応用することを検討し，試行を行ってきた．

本章は，日立製作所情報グループのプロジェクトマネジメントセンタが主体になって推進してきた活動をまとめたものである．ＩＴシステム開発プロジェクトの特徴をまとめると共に，伝統的プロジェクトマネジメントの問題点を考察し，リスクマネジメント，スコープマネジメントを中心としたモダンプロジェクトマネジメント適用の試行とその評価を試みる．

6.2　ＩＴ開発プロジェクトの特徴

ＩＴシステムの特徴は，その構築中，構築後を含めて全体像を直接目にすることができないということである．例えば古くは，巨大な建造物であるピラミッド

や万里の長城，現代では，ビルや架橋などは，もちろん専門家でなければその建造物の品質などを，認識・評価できない．しかし，その建設の進捗は，素人目にも分かる部分がある．

ＩＴシステムは，極端な言い方をすれば，全て電子的に作成し，アウトプットとして形あるものは一切なしという構築方法も可能である．この場合，開発中のシステムも，開発の終了したシステムもコンピュータの中に記録されているだけになる．

また日本においては，ＩＴシステム開発において，重要な位置づけとなる業務の流れやデータの流れなども，ドキュメント化やマニュアル化が完璧になされているケースは少ない．この傾向は，米国と比較して顕著であると考える．

「以心伝心」とか「阿吽の呼吸」とかいうのがあって，お互いの分担（スコープ）をとことんつきつめるという文化は，日本にはなかった．このために，顧客とベンダ間での分担への思いが食い違い，大きな隙路となる場合がある．

またプロジェクト内でも，短期間に数百人規模の要員が必要になることもままあり，関連会社などの要員を充当して開発プロジェクトを構成することがほとんどである．そのため，プロジェクト内でのスコープ，原価，工程，品質，コミュニケーション，ヒューマンリソースなどが見えにくくなる心配がある．

最近のいくつかのＩＴシステム開発プロジェクトでは，この問題を解決するためにプロジェクトマネジメントプロセスの適用とツールの利用を推進している．これからのＩＴシステム開発プロジェクトでは，これらのツールや手法をシステムとして統合した，ＣＡＰＭＳ（Computer Aided Project Management System）が必要になってくると考えられる．

6.3 伝統的プロジェクトマネジメントの問題点

伝統的プロジェクトマネジメントは，ＫＫＤ（経験・勘・度胸）で代表されるように，主観的・体験的であるので，同じ体験をした者だけが理解し合えるものであった．モダンプロジェクトマネジメントでは，このＫＫＤにもう１つのＫ（知識）を加えることによって，客観的・理性的で，他者や未来の人にも伝達できる形式知を基本にしている．

このＫ（知識）は，ＰＭＢＯＫ[9]によれば，９つのエリアに分類できる．前述の「原価管理(Cost Management)」，「工程管理(Time Management)」と「品質管理（Quality Management)」を除く６つの知識エリアが，伝統的プロジェクトマネジメントにはない．これによるＩＴシステム開発プロジェクトにおける問題点を列記する．

（１）スコープ管理(Scope Management)が無い

・顧客への約束（何をすれば満足して頂けるのか）が不明確

・顧客，コンサルタント会社，他ベンダとの間での責任が不明確（グレーゾーンが下流工程での軋轢を生む）
・システム開発規模拡大に対する危機意識が薄い

（2）調達管理(Procurement Management)が無い
・工程管理または原価管理からの派生的事務処理としての取扱い
・マネジメントによる調達品質の確保という行動の欠如

（3）コミュニケーション管理(Communication Management)が無い
・各ステークホルダの情報ニーズに無頓着かつ思いつきで対応
・プロジェクト協調者を増やすための意識希薄
・プロジェクト内での各種情報共有と伝達が非効率

（4）組織管理(Human Resource Management)が無い
・形式的に体制図をつくり，ファイリングし，現実と乖離
・RAM（Responsibility Assignment Matrix）表が無いのでメンバの責任と権限が曖昧
・中長期的人材育成のストーリー無し

（5）リスク管理(Risk Management)が無い
・リスクの共通認識の欠如（プロジェクトマネジャのみ不安感をもつ）
・断片的で非体系的なリスクへの対応
・各リスクへの対応優先度無し
・先手管理意識が希薄

（6）統合管理(Integration Management)が無い
・実現可能なプロジェクト計画を立てない，建前の計画書を作りファイリング
・すぐ物作りに突入
・プロジェクトマネジャがどのようにマネジメントし，どのように品質，性能，機能を確保して行くのかの指針が不明確

このような伝統的プロジェクトマネジメントの問題点を認識して，モダンプロジェクトマネジメントの適用を検討した．

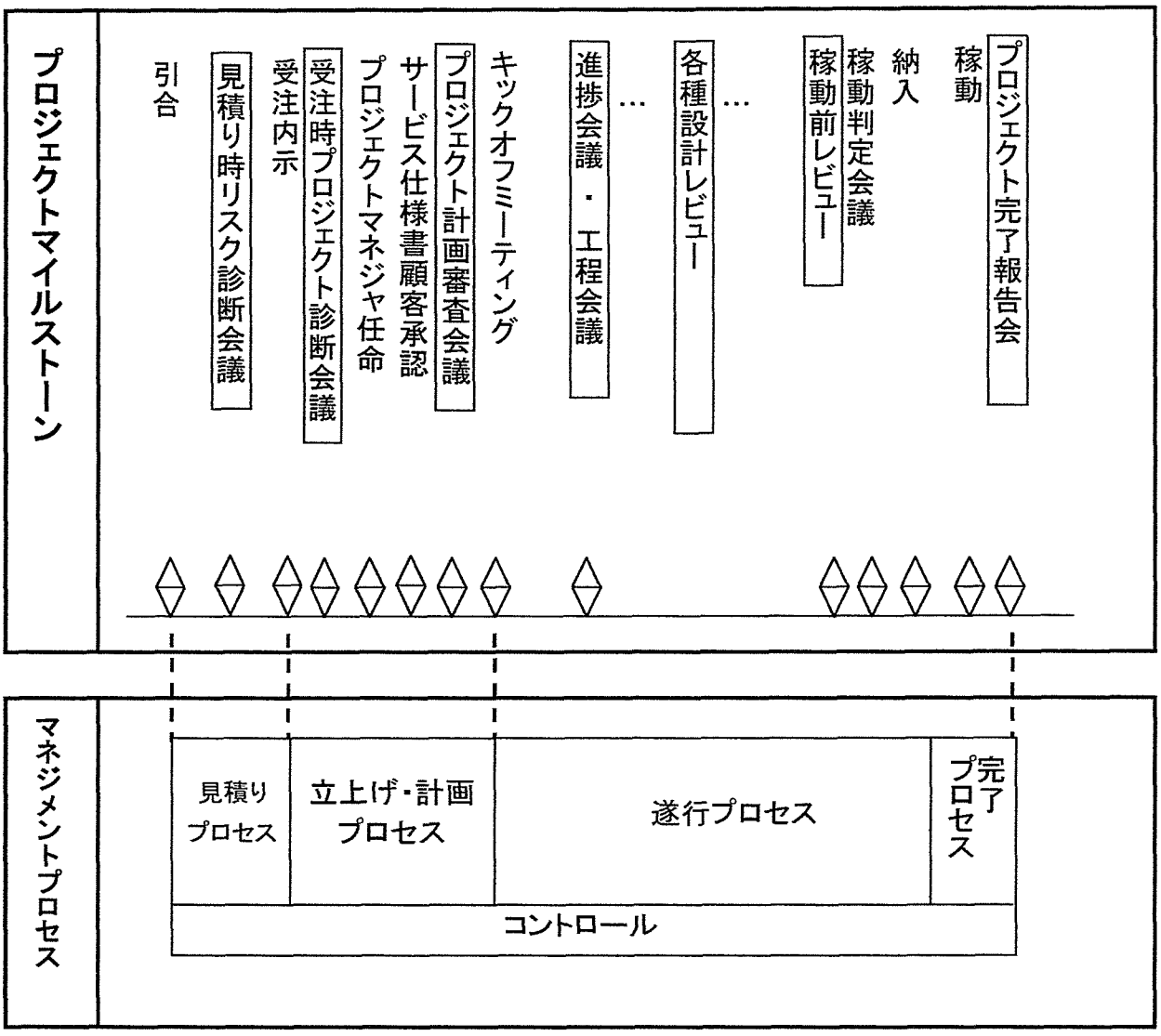


図 6.1　プロジェクトのマイルストーンとマネジメントプロセスの概念図

6.4　モダンプロジェクトマネジメント適用の検討

ＩＴ開発プロジェクトにモダンプロジェクトマネジメントを適用する必要性は認められたが，適用へのアプローチは平坦ではない．制度の確立，組織の編成，プロジェクトマネジャおよびプロジェクト構成メンバの知識習得とモチベーションの高揚などの１つ１つを推進してきた．また，ＩＴ開発プロジェクトで，どの知識エリアから適用するかを検討するのもポイントであった．

モダンプロジェクトマネジメントのどの知識エリアから重点的に適用するかについては，いわゆる「混乱プロジェクト」や「失敗プロジェクト」の事例調査を踏まえて決定することにした．

その結果，調査対象としたプロジェクトでは，特にリスクマネジメントとスコープマネジメントの強化が必要であることが判明した．リスクマネジメントについては，そのリスクが表面化してから対策に乗り出す傾向が強く，問題処理型のプロジェクトマネジメントになっていた．この解決策として「リスク早期抽出自己診断表（略称 PM-PAS*）*」を作成した．

また，スコープマネジメントの強化策としては，顧客との間でスコープを明確にするための「ソリューション仕様書」の作成と顧客承認を義務付け，プロジェクトマネジメントを側面から支援するスタッフ部門から，その進捗をフォローすることとした．

もう１つは，それらのプロジェクトマネジメント施策をプロジェクトに適用するタイミング（工程）を検討した．対象プロジェクトのマイルストーンとマネジメントプロセスの概念を，図 6.1 に示す．

PM-PAS については，「見積り時リスク診断会議」前に自己診断し，その会議への提出を義務付けた．その後，「受注時プロジェクト診断会議」，「プロジェクト計画審査会議」，「工程会議」，「稼動前レビュー」などの前に自己診断することを義務付けた．

「ソリューション仕様書」については，「見積り時リスク診断会議」に素案を提出することにした．ただし，この時点では，スコープが不明瞭なケースが多く，次の「受注時プロジェクト診断会議」に顧客提出版を提示することを義務付けた．

6.5 モダンプロジェクトマネジメント適用の具体例

モダンプロジェクトマネジメントの適用について，施策の具体例２項目について，その概要を記述する．

（１）リスク早期抽出自己診断表（PM-PAS）

PM-PAS は制定以来，改版を重ねており，現在のチェック項目数は，上流工程分で１６６項目となっている．表６．１は，リスク自己診断項目を診断ポイントと知識エリアで分類したものである．横軸にプロジェクト工程を取り，縦軸に知識エリアを取っている．横軸については，初期の４つの診断ポイント「要求分析時」，「提案・見積時」，「受注時」，「計画時」を取り，下流工程分は割愛している．

チェック項目数としては，ＩＴ開発プロジェクトでマネジメントを強化すべきと判断した「スコープ」に関してが７３項目と最も多く，また，どの診断ポイントでもチェックされている．ついで多いのが，伝統的プロジェクトマネジメント

＊PM-PAS：Project risk Management Pre-self Assessment Sheet

の３項目「原価」,「工程」,「品質」で,「原価」については提案・見積り段階での診断項目が多く,「工程」,「品質」に関しては「計画時」にチェックされている項目が多い.

表 6.1　自己診断項目分類表

No.	診断ポイント / 知識エリア	要件分析時	提案･見積時	受注時	計画時	計
1	スコープ	12	**33**	12	16	**73**
2	原価(コスト)		8	5	2	15
3	組織	1	2	2	9	14
4	調達		1	3	4	8
5	コミュニケーション			3	8	11
6	工程(タイム)		3		16	19
7	品質	3	1	2	10	16
8	統合	1	2	2	5	10
合計		17	50	29	70	166

（2）ソリューション仕様書

ソリューション仕様書には，作業項目別の分担を記述すると共に，システム化対象範囲，業務要件（機能一覧，性能要件，信頼性要件など），システム構成，移行の考え方，顧客から提示される資料等，納入物，作業場所，再見積り条件などを記載する．また，ソリューション商品内訳と契約形態（委託，請負など），検収方式を記載する．

プロジェクトマネジャは，これを顧客に承認して頂くまで，プロジェクトマネジメント支援部門からフォローされる．

6.6　今後の課題

対象とするＩＴシステム開発プロジェクトに対するモダンプロジェクトマネジメント適用の一端について述べてきたが，更に，この流れを定着させると共に拡充してゆきたいと念願している．そして，「属人的プロジェクトマネジメント」から「組織的プロジェクトマネジメント」へ,「問題処理型」から「計画重視・先手管理型」への転換を図って行きたい．

このような，定着と拡充には，プロジェクトを構成する要員の教育が不可欠である．教育については，長年ＫＫＤ（経験・勘・度胸）でプロジェクトマネジャ

を務めてきたベテランと，そして中堅，これからプロジェクトマネジャになる比較的若手の要員の３つに分けて，知識教育や演習などを行っている．

また，これら教育の受講実績に加えて，公的資格の有無，プロジェクト経験などを加味した，「プロジェクト規模別のプロジェクトマネジャ資格認定制度」を実施している．

ＩＴシステム開発プロジェクトを囲む環境は，これまでにもまして激しく変化することが予想される．ＩＴシステム開発プロジェクトを成功させるために，本章で述べた制度や教育の更なる拡充を課題として取り組んで行きたい．

6.7 まとめ

ＩＴシステム開発プロジェクトのマネジメントには，モダンプロジェクトマネジメントの知識体系の適用がますます有効になってくると思われる．ＩＴ分野固有の課題を解決すると共に，ＩＴ分野のプロジェクトマネジメントに共通の知識を集約して活かす道を見つけて行きたい．

第７章 仕様確定状況の監視支援－仕様発散防止 QFD

7.1 はじめに

ソフトウェア受託開発では，当初設定した予算や納期に収めることができない，テスト段階や顧客への納品後に品質上の重要な欠陥が発見される等の深刻な問題をもつプロジェクトが発生することがある．これらの主たる原因は下記の２つである．

①見積りミスによるコスト超過

②不適切なプロジェクト管理

これは，昨今の顧客をとりまくビジネス環境の変化の速さ，ＩＴの進歩の速さに起因することは言うまでもないが，より本質的な問題として，ソフトウェアの性質に起因するところが大きい．ソフトウェアは無形物であるため，プロジェクト初期の段階では仕様を明確にしにくく，プロジェクト進行とともに少しずつ仕様が明確になっていくという特徴をもつ．そのことが，見積りミスを誘発したり，プロジェクト管理を難しくしている．

上記課題を解決するには，下記の方策を取ることが必要である．

（１）ソフトウェアの仕様や制約，プロジェクトの作業内容，顧客との作業分担等，コストを左右する部分の要件や前提条件を見積り段階で明確にすること．

（２）（１）の変更を常に追跡し，コストやスケジュールへの影響を把握し，その影響度に応じた適切な措置をとること．

追跡すべき項目として特に重要なのがソフトウェアの「仕様」である．仕様変更が多発すると必然的に開発作業量も増加し，コスト増に直結する．従って，ソフトウェアの「仕様」の確定状況を監視し，確定していない仕様については早急に確定させるような対策を講じることがソフトウェア受託開発においては極めて重要である．この仕様の確定状況監視のために，日立製作所情報グループ生産技術本部が主体となって，QFD*（品質機能展開）[24]，[25]，[26]を応用した「仕様発散防止 QFD」と呼ぶ技法を開発した．本章では仕様発散防止 QFD の概要と有効性について述べる．

---

* QFD（Quality Function Deployment,品質機能展開）とは，日本で考案された品質管理技法である．システムに対するユーザの要求品質を分析・系統化し，これを代用特性に転換して設計品質・仕様を定め，系統的に展開する．

7.2 QFD の適用の狙い

QFD（品質機能展開）は，顧客・市場の声を設計・製造といった技術的手段へ変換するための技法である．QFD を適用することにより，

①採用すべき技術的手段の決定過程が定量化・明確化・可視化できる．

②広い範囲の対象の中から，採用すべき技術的手段の重点化ができる．

という効果を得ることができる．

ソフトウェア受託開発プロジェクトでは，ソフトウェアの備えるべき機能に関する仕様の発散・未確定を早期に検出し，手戻りに起因するコストアップを防止することが重要である．そのためには，受託開発するソフトウェアの備えるべき各機能について，仕様の確定状況を監視し，機能の重要度や複雑度を加味して，対策の要・不要の判断や対策すべき機能の順序付けを行う必要がある．これを実現する技法が，QFD を応用した仕様発散防止 QFD である．

7.3 仕様発散防止 QFD の概要

仕様発散防止ＱＦＤでは，仕様確定の遅延への対策の必要がある機能の特定や，対策の優先度をつけるために，各機能に対し「対策要求度」と呼ぶ指標を定義する．対策要求度は，機能のニーズを表す重要度と，機能実装の難しさやシステム全体に及ぼす影響を表す複雑度をもとに算出される．

機能の対策要求度は，下記のように算出する．

①機能一覧を作成する．
②顧客ニーズを基に機能の重要度を設定する．
③機能複雑度を，仕様確定遅延時や仕様変更発生時の他への影響の観点（n 項目）から評価する．
④機能の対策要求度を，機能複雑度に，仕様の発散・不確定状況の観点（m 項目）からの評価を加味して，算出する．

①～③は最初に実施する作業，④は定期的（1 週間～2 週間ごと）に仕様が確定するまで実施する作業である．

図７．１は，仕様発散防止 QFD の全容を示したものである．

（１）機能重要度の算出

機能の重要度を算出するには下記の作業を行う．

①ニーズ一覧の作成

②ニーズの重み付け

③機能とニーズの付き合わせによる機能の重要度の算出

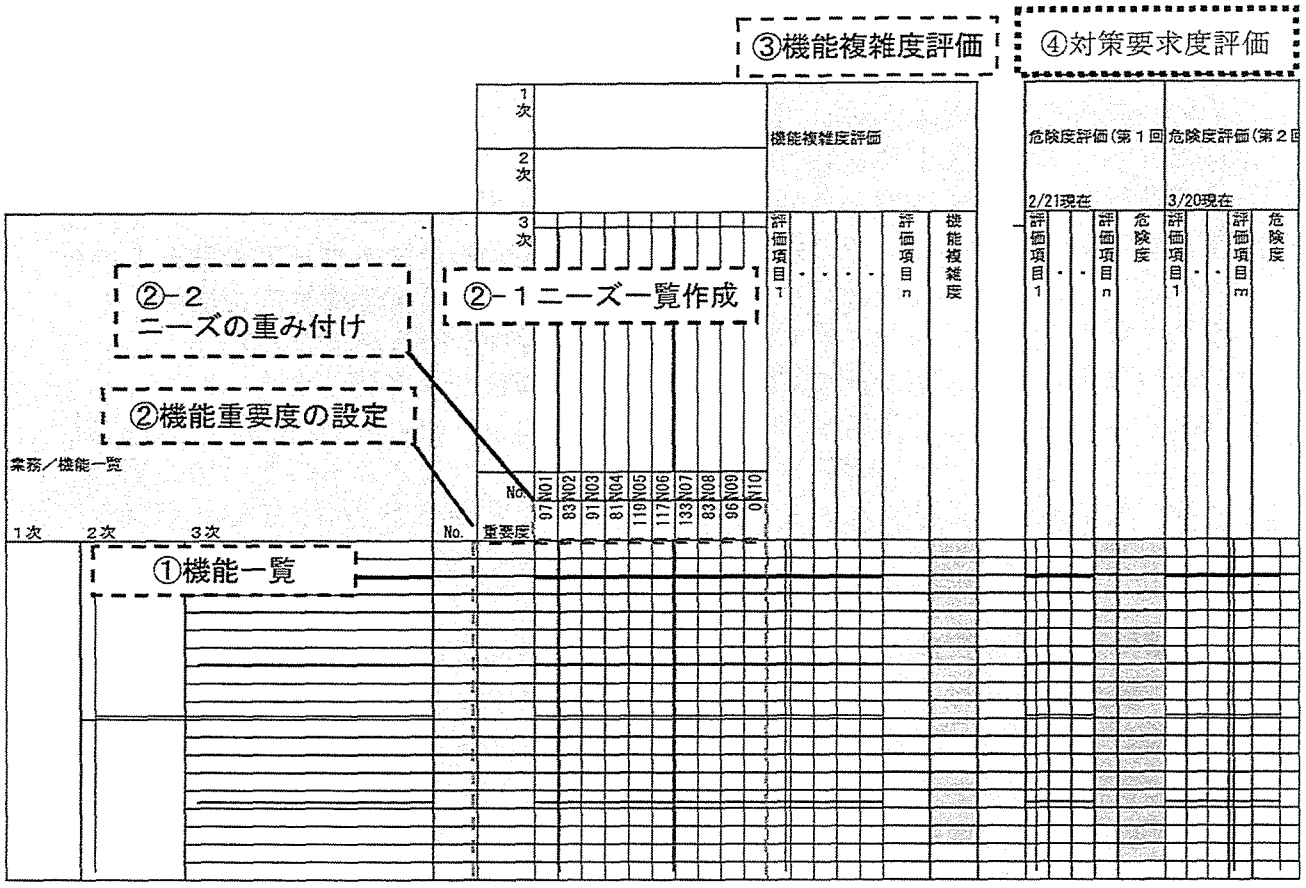


図７．１　仕様発散防止 QFD の成果物

まず，「②－１ニーズ一覧の作成」と「②－２ニーズの重み付け」を行う．顧客がどのニーズを重視しているかを評価し，その評価をもとにニーズの重み付けを算出する．

次に，あらかじめ作成した機能一覧と，今作成したニーズ一覧表をもとに機能とニーズの関係評価を行い，機能の重要度を算出する．機能の重要度の算出方法は，ニーズの重要度と機能・ニーズ間の対応強度をもとに，独立配点法で算出する．　この重要度は顧客ニーズとの関連性の高さを示すもので，重要度の高い機能ほど顧客からの注目度が高いと考えられる．

（２）機能複雑度

各機能の複雑度を，ｎ個の評価項目を評価し算出する．機能複雑度は，その機能が持つ難しさを示した値であり，機能の特性やシステムインタフェースの有無

などで評価する.

n個の評価項目のウェイトを乗算することにより,次式の機能複雑度を求める.

$$(\text{機能複雑度}) = (\text{機能重要度}) \times \underbrace{(\text{第1ウェイト}) \times \cdots \times (\text{第nウェイト})}_{\text{機能複雑度係数}}$$

（3）対策要求度の算出

各機能の対策要求度を，m個の評価項目を評価し，次式により算出する.

$$(\text{機能対策要求度}) = (\text{機能複雑度}) \times \underbrace{(\text{第1ウェイト}) \times \cdots \times (\text{第mウェイト})}_{\text{機能対策要求度係数}}$$

7.4 仕様発散防止 QFD の活用方法

各機能の対策要求度により，ソフトウェアの備えるべき多くの機能の中で仕様を早く決定すべき機能はどれかを把握したり，仕様が収束していかない機能を検出することができる.

機能の対策要求度が高いという現象は，その機能に何らかの対策が必要であるという警告であり，対策内容は，プロジェクトの状況により異なる．この警告を契機に，管理者は，担当者からのヒアリング等により問題の根本原因を把握し，対策する必要がある.

（1）対策要求度が高い場合

作業者のスキルを含め，現状体制で十分かどうか確認する.

（2）対策要求度の推移が思わしくない

対策要求度の判定を複数回実施した場合，前回より上がっている，下げ幅が小さい，変化がない，などの問題が浮かび上がる．これらに対して，原因を至急究明することが必要である．体制，スキル，顧客などに何らかの問題がある可能性が高い.

問題数の推移（異常に増加，解決が非常に少ない等），進捗報告と比べて矛盾がないか，にも着目する.

(3)品質表が作成できない

機能一覧が作成できない，機能複雑度や対策要求度を算出するための評価項目の評価ができないプロジェクトでは，プロジェクト管理ができていない，実態を

把握している人がいないという問題がある.また,プロジェクトの初期段階では,品質表を作成できない場合もあるが,機能一覧,他システムとの関係,仕様提示度合,問題解決状況を明らかにし,できるだけ早期にこの品質表*を作成することが必要である.

7.5 適用効果

仕様発散防止 QFD の適用により,以下の効果がみられた.

(1) 的確な対策優先順位付け

仕様発散防止 QFD を適用した複数のプロジェクトについて,担当者へのアンケートで「機能複雑度や対策要求度は,その時点でプロジェクトの主要な問題がどこに潜んでいるかを的確に示したと思うか」という問いに対し,「そう思う」と回答した割合が 90%に達した.また,定められた納期を達成できなかったプロジェクトは 30 件中 2 件で 6 %程度しかなかった.これらの事実から,仕様発散防止 QFD が的確な対策優先順位付けに効果を発揮し,結果として納期遅延を防いでいることがわかる.

(2) 可視化

プロジェクト全体をいくつかの機能に分類し,それらの状況を A3 程度の大きさにまとめることで,プロジェクト全体の状況把握が容易に行えるようになり,問題箇所の発見がスムーズに行えるようになった.この点は,特にプロジェクト管理者や,より上位の管理者から高い評価を得ている.

(3) 適用効果の小さかったケース

アンケートで「機能複雑度や対策要求度は,その時点でプロジェクトの主要な問題がどこに潜んでいるかを的確に示したと思うか」という問いに,「そうは思わない」と回答したプロジェクトが 10%あり,調査した結果,下記の 2 つの場合であることが判明した.

(a) どの機能も対策要求度が高い場合
(b) 認識どおりの結果である場合

(a) の場合は,特定部位の対策でなく,全体的な根本対策が必要であることを示しており,また (b) の場合は,スキルの高くない技術者,管理者でも仕様

---

*品質表:通常,縦の見出しに要求品質(顧客ニーズ)を列挙し,横の上段に代用特性(製品の機能)を並べ,両者の対応を表現した 2 元表である.

発散防止 QFD を適用することで，同様の認識をもつことができ，効果は大きいと判断できる．

7.6 まとめ

仕様発散防止 QFD は，定量化，可視化，重点化指向といった QFD の特徴を生かし，QFD の適用が進まなかったプロジェクト管理の分野にも適用でき，非常に効果の大きいものであると評価できる．今後もこの方式を発展させていくとともに，これを機にプロジェクト管理の定量化，可視化，重点化指向を進めて行きたい．

第８章　レビュー品質向上手段の導入による設計品質の向上

8.1　はじめに

ソフトウェアの品質向上の鍵は設計品質の向上にある．なぜなら，設計の良否はそれ以後の開発工程に大きく影響し，設計工程で作り込まれた不良が後工程で検出されるほど，修正コストが増加し，生産性が低下するためである．加えて，長期的にとらえても，設計の良否は保守性や拡張性を左右する．以前から，設計品質を向上するための中核技術として，設計レビューの重要性が唱えられていた．設計レビューにより，設計の品質を評価し，設計内容にフィードバックすることは，各企業で各々のやり方でなされてきたことである[27]．

設計の品質は設計レビューのあり方によって決まるとの前提にたって，設計レビューの出力であるレビューの議事録に着目した．レビュー議事録には次の４つの特徴がある．

(1)結論だけでなく，結論に到った経緯が記述されている．
(2)採用された案だけでなく，棄却された案も記述されている．
(3)だれが何を発言したかが具体的に記録されている．
(4)レビューの対象となった資料や議論の材料になった参考文献への参照先が記録されている．

日立製作所情報グループ生産技術本部では関連部署との連携のもとに，上記の特徴をもつレビュー議事録を用いて設計の品質評価を行うこととして，レビュー議事録の検索や分析機能を備えたツールを開発した．本ツールの特長は，プロジェクトマネジャが設計の充実度を定量的に評価できること，および設計の良否を定性的に評価できることにある．

本章では，はじめに，設計の品質向上がレビュー品質の向上により可能であることと，レビューの品質がレビューの議事録によって管理できることを説明する．次に，レビュー議事録を使って設計の品質を管理する方法と，それに必要な機能を説明する．最後に，本取り組みの適用状況と適用効果について述べる．

8.2　設計レビューの議事録による設計の品質管理

8.2.1　ソフトウェア開発工程におけるレビューの位置づけ

ソフトウェア開発工程とレビューとの関係を図８．１に示す．レビューは，設計工程の成果物を評価し，技術的な問題を解決することにより品質を作り込む作業である．成果物の評価を誤ったり，技術的問題の摘出が不十分であったり，摘出できてもその解決が適切になされなかった場合，設計工程で不良が作り込まれる[28]．テストは，プログラムを評価して品質確認を行う作業である．設計工程で不良が作り込まれていれば，不良を摘出する．

このように，ソフトウェア開発の全工程を通して，レビューとテストによって品質保証を行う．しかし，テストは不良が作り込まれている場合にそれを摘出する作業であり，設計の品質向上にとっては，設計工程において不良が作り込まれないことが大事である．こうした点から，レビュー自体の品質向上は重要である．

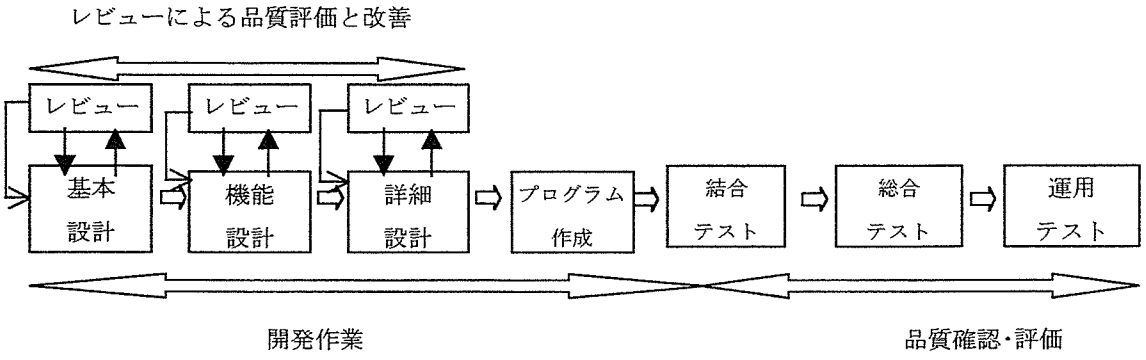


図８．１ ソフトウェア開発工程におけるレビューの位置づけ

8.2.2 レビューの実施方法

レビューの実施方法は，多種多様である[27]．本論文で提案するレビューによる設計の品質管理を理解しやすくするために，我々が実施しているレビューについて，目的，実施のタイミング，報告書の観点から説明する．

（１）目的

レビューを実施する目的には，技術面と管理面の２面がある．

技術面のレビューでは，①設計の妥当性の確認，②技術的問題の解決，が主目的である．①では，誤りや後工程で問題になると予想できる点を早期に摘出し，回避するように努める．②では，新技術を採用したり，技術的に難しい問題を抱える場合に，レビュアの意見を求め，問題の解決を図る．問題の摘出と解決のいずれにおいても，レビューメンバーには技術力や経験が必要であるため，レビューが成功するには，必要な技術に関する有識者や，経験者がレビューメンバーに入っていることが重要になる．

管理面では，作業の進捗状況および品質状況を把握することである．技術的な問題を発見した場合に，その技術的な内容よりも，問題解決の分担といった次に起こすべき行動を検討する．レビューで的確な判断を下せるためには，進捗や品

質の状況を把握できるための情報が必要である．

（２）実施のタイミング

技術面のレビューでは，工程の終了時点を区切りとして実施するものと，レビューを必要とする事情が生じて行うものの２種類がある．前者では，設計書などの成果物が一応完成した時点で，品質保証部門を交えて成果物をチェックする．後者では，設計の過程で詳細を詰めていくために，有識者や経験者を交えて，設計内容を決める．設計の詳細は，性能要件，コスト，使い勝手などの点から全体のバランスをとりながら決定する必要があるため，こうしたレビューが必要になる．レビューでは一度に大量に議論はできないため，レビューは頻繁に行う．特に，大規模システムの設計では，こうした小さなレビューの積み重ねにより，設計は徐々に完成していく．

（３）レビュー内容

レビューでは複数のテーマが検討されることが多い．技術面のレビューでは，(2)で説明したように，設計の詳細は，性能要件，コスト，使い勝手などの点から全体のバランスをとりながら決定する必要があるため，関連性のあるテーマ（たとえば，システム処理方式設計とネットワーク構成設計など）も合わせて議論する．

レビューで検討する内容は，設計の成否を左右する重要なことが多い．レビューでの検討が不十分であったり，結果が適切でなかった場合には，開発するシステムの品質が低下したり，開発中あるいは稼動後に問題が生じる．従って，レビュー内容をプロジェクトマネジャはチェックする必要がある．レビューに出席できなかった場合には，（４）で説明する「報告書」のチェックにて行う．

（４）報告書の形式

レビューを実施した後には，レビューの結果・実績の記録としてレビュー議事録を残す．

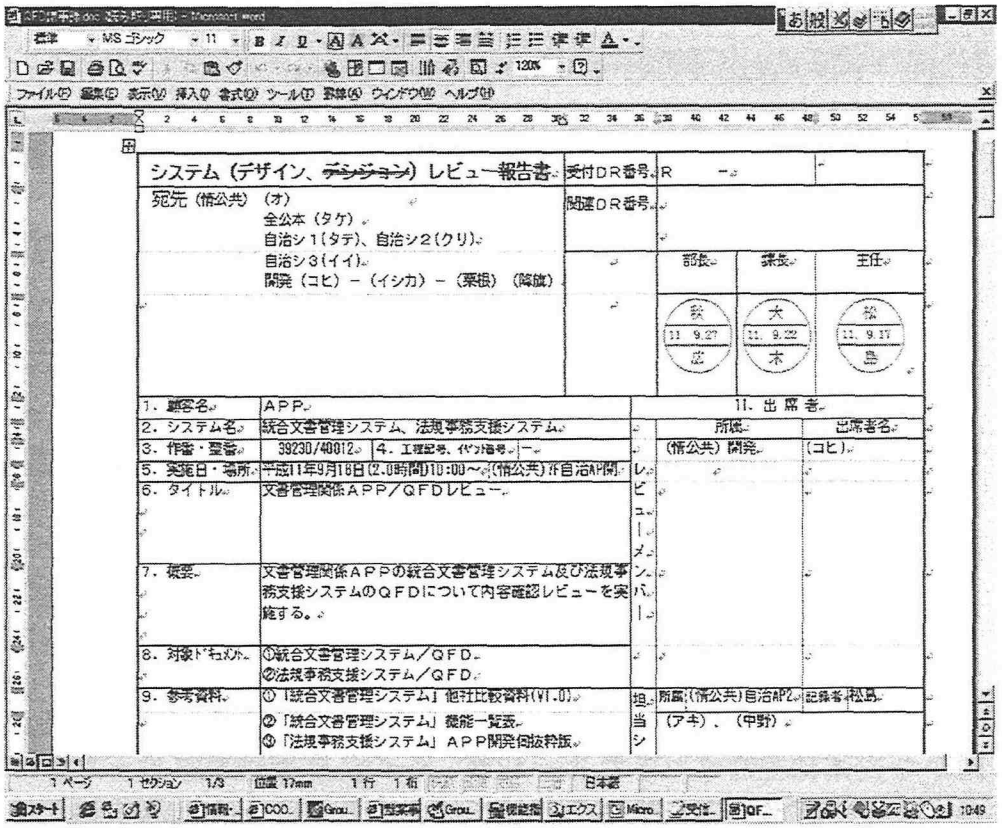
我々の用いているレビュー議事録の様式を図８．２に示す．

レビュー議事録は表紙と次紙から構成される．

表紙には，レビュー日時・出席者(所属と氏名)・レビューテーマ，概要，成果，今後の課題，レビュー対象の資料およびレビューの材料となった参考文献などを記載し，レビューの概要が把握できるようになっている．

次紙には，検討テーマごとに，結論とその理由とを記載する．結論の理由とは，どのような案を，どのような視点から比較検討したのか，何が決め手となって結論の案を選択したのか，それは資料や参考文献のどこに基づいた考えであるか，などであり，議論の詳細と言える．結論のみならず，理由を記載することにより，レビュー議事録を見れば設計の考え方や方針が把握できる．

従って，開発中の設計を診断する場合や，開発中に問題が生じて設計を見直す場合には，レビュー議事録は重要な参考資料となる．



システム（デザイン、~~デバッグ~~）レビュー報告書

受付DR番号	R　－
関連DR番号	
宛先（情公共）（オ） 全公本（タケ） 自治シ1（タテ）、自治シ2（クリ） 自治シ3（イイ） 開発（コヒ）－（イシカ）－（栗根）（降旗）	部長：鈴木 11.9.27 広 課長：大久保 11.9.22 本 主任：松島 11.9.17 島

1．顧客名	APP	II．出席者	
2．システム名	統合文書管理システム、法規事務支援システム	所属	出席者名
3．作番・整番	39230/40012	（情公共）開発	（コヒ）
4．工程記号、イベント番号	－		
5．実施日・場所	平成11年9月16日（2.0時間）10:00～（情公共）自治AP開		
6．タイトル	文書管理関係APP／QFDレビュー		
7．概要	文書管理関係APPの統合文書管理システム及び法規事務支援システムのQFDについて内容確認レビューを実施する。		
8．対象ドキュメント	①統合文書管理システム／QFD ②法規事務支援システム／QFD		
9．参考資料	①「統合文書管理システム」他社比較資料(V1.0) ②「統合文書管理システム」機能一覧表 ③「法規事務支援システム」APP開発伺抜粋版	担当システム	所属：（情公共）自治AP2　記録者：松島 （アキ）、（中野）

図８．２　レビュー議事録の形式

8．対象ドキュメント	①統合文書管理システム／QFD ②法規事務支援システム／QFD		
9．参考資料	①「統合文書管理システム」他社比較資料(V1.0) ②「統合文書管理システム」機能一覧表 ③「法規事務支援システム」APP開発伺抜粋版 ④「法規事務支援システム」H11／下構造項目資料	担当システム	所属：（情公共）自治AP2　記録者：松島 （アキ）、（中野）
10．件数	検討　4　件　対策　2　件　持越　3　件		

12．今後の検討項目	項目	担当部署	目標期日
	(1)統合文書管理システムQFDの指摘事項修正の実施	自治AP2	H11/9/24
	(2)統合文書管理システム他社比較資料のメンテナンス実施	自治AP2	H11/9/30
	(3)法規事務支援システムQFDの指摘事項修正、及び将来開発項目の追加	自治AP2	H11/9/24

13．主な成果	
	(1)各QFDの
	(2)各システ　項目が明確になった。

図８．３　レビュー議事録を書く意義

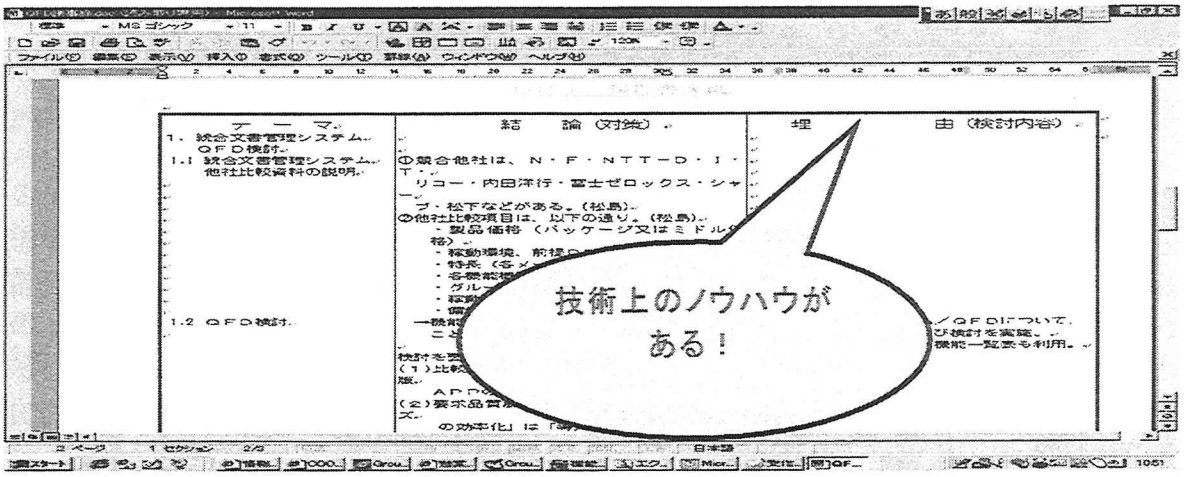


図８．４　レビュー議事録の技術上のノウハウ

8.2.3　レビュー議事録によるレビューの品質管理

レビューの良否は出席者の技術力に依存する．つまり，レビューの品質は人という不確かな要素に左右されるため，レビューの品質管理が必要となる．

１つの方法としては，どのような根拠で設計したか，どんな案を比較検討したのか，どのような資料や参考文献を元にしたのかといった視点から定性的に管理する方法がある．これらは，８．２．２で説明したように議事録の次紙の結論欄とその理由欄，表紙にある参考文献や参照元から容易にわかる．

もう１つの方法は,以下の視点からレビューを定量化して管理する方法である．

(a)有識者や技術力のあるメンバで議論したか．

(b)議論は十分か．

(c)議論した内容は設計全体を網羅しているか．

このうち，(a)については，議事録の表紙に記載された出席者の氏名と所属から容易に判断できる．(b)は，同じテーマを何回にわたって議論しているかという視点で押さえられる．８．２．２の(2)で述べたように，大規模システムの場合，設計はレビューの積み重ねにより完成度が増していく．例えば，データ設計という一つのテーマについても，通常何回かのレビューを実施するので，レビュー実施の回数が，議論の程度を知る指標になる．(c)については，ソフトウェア品質特性のうち，レビューを実施したものは何か，していないものは何かを調べることで判断できる．これは，レビュー議事録の次紙全体をテーマごとに分析することにより把握できる．

「レビュー議事録からレビューの品質が把握できる．さらに，レビューの品質管理を行うことで設計の品質管理ができる」という仮説を立て，ツールの作成を試みた．

8.3 設計の品質管理に必要な機能

8.3.1 設計の良否の評価機能

設計の良否の評価は，設計の考え方や方針をチェックすることにより行う．設計の考え方や方針は，レビュー議事録に議論の詳細として記述されている結論とそれに到った理由の記述からおさえることができる．評価するには，データ設計とか，処理方式設計といった特定の設計項目ごとに検討内容を把握するのが効果的である．

設計レビューは，１つのテーマが複数回のレビューにまたがって議論され，１回のレビューでは，複数のテーマについて議論されるという特徴がある．従って，１つのテーマに対する議論が複数のレビューの議事録に分散しており，１つのレビューの議事録には複数のテーマに対する議論が混在して存在する．そのため，ある設計項目に関する議論の詳細を把握したい場合には，複数のレビュー議事録を参照しなければならない．

そこで，レビュー議事録から議論の内容を切り分け，それを見やすい形式に編集して取り出せる機能が必要である．本ツールでは，参照したいテーマについて記述された議事内容のみを抽出し，時系列的に表示する機能をもたせた．こうして，誰の発言により，どのような経緯で仕様が決定したのかを知ることが可能になり，検討の経緯やコンテキストを容易に把握できるようにした．

8.3.2 設計の充実度の評価機能

設計の充実度の評価は，レビューしたテーマ数とテーマごとのレビュー実施回数をチェックすることにより行う．

設計レビューは，ソフトウェア品質特性（ISO 9126,1991）を向上するために実施するものである．ソフトウェア品質特性を網羅するように，あらかじめシステム設計時にレビューすべきテーマの一覧表を用意する．これを標準テーマと呼ぶ．

充実度の評価は，レビューの議事録に記述されたテーマと標準テーマとを照合し，各テーマに対するレビュー実施回数から判断する．そこで，本ツールでは，各テーマに対するレビュー実施回数を一覧表示する機能をもたせた．

レビュー実施回数が少ない場合は，議論が不十分である可能性，逆にレビュー回数が多い場合は議論が収束しない，あるいは無駄なレビューを実施している可能性がある．また，レビュー実施回数が０のテーマについては，検討漏れの可能性がある．本ツールでは，レビュー実施回数の一覧表示画面上に，テーマ全体に占めるレビュー実施回数が１以上のテーマの割合を数値で示すことにより，検討

の網羅度を示すようにした．

標準テーマについては，プロジェクトで自由に編集できるようにカスタマイズ機能を提供した．プロジェクト開始時に標準テーマをベースとして，プロジェクト内でレビューテーマを選定し，レビュー計画を立てることで，効果的な設計レビューを実現する．

8.4 ツールの実装

8.4.1 システム構成

ツールは，WWWサーバとＣＧＩプログラム，リレーショナルデータベースシステム（ＲＤＢＭＳ）を用いて実装した．利用者は，WWWブラウザを通してシステムを利用する．組織内のすべてのレビュー議事録と添付ドキュメントはサーバで一元管理される．

8.4.2 レビュー議事録の入力と情報の出力

レビュー議事録をデータベースに格納する際は，レビューの概要である表紙の情報と，議事内容である次紙の情報とを各々別のテーブルに格納する．議事内容の情報は，指定された情報のみを抽出できるように，テーマ単位に分割し，個別のデータ・レコードとして格納する．

このように格納された議事レコードに対して検索を行うことで，ヒットした議論の議事内容を抽出する．検索キーワードを「ＤＢ物理設計」とした場合の検索結果の例を図８．５に示す．この例では，３つのレビュー議事録からそれぞれ該当する議事内容が抽出されている．抽出された議事内容は，レビュー実施日でソートされて表示されるため，議論の経緯を追うことができる．

図８．６は，上記のような議事レコードのテーマと標準テーマとを照合し，テーマごとのレビュー実施回数を出力した結果である．レビューした項目に追加して，標準的にレビューしなければならない項目，あるいはプロジェクト開始時に決めたレビュー項目との過不足を表示することで，設計の充実度を定量的に把握できる．

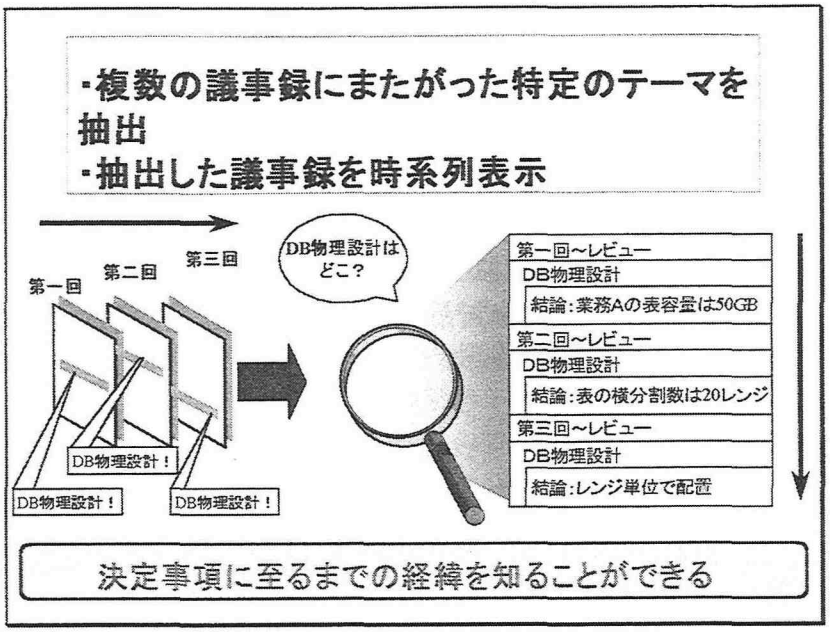


図８．５ 複数議事録にまたがる特定事項の抽出と時系列表示

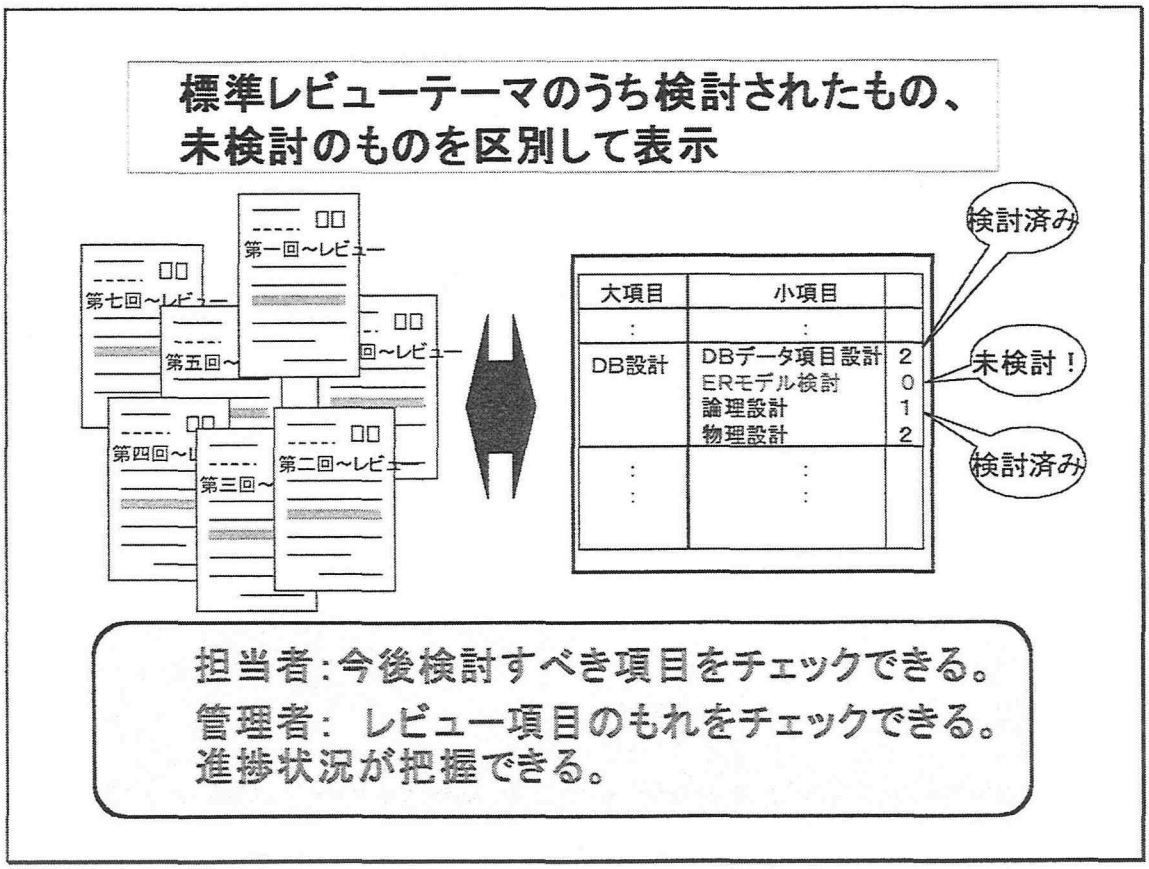


図８．６ 標準レビューテーマと検討項目の比較によるレビュー充実度の評価

8.5 適用評価

8.5.1 適用状況

1999年6月に適用を開始後，2度の機能拡張を行い，現在，日立製作所内で，800の業務ソフトウェア開発のプロジェクトで適用している．これらのプロジェクトは，官庁や地方自治体，大学，病院，金融機関，流通・サービス業など多岐にわたる業種の財務会計，税金，給与といった基幹業務およびユーザ業務用の特注システムを開発する大規模プロジェクトから，社内の事務や開発支援のツールを開発する小規模プロジェクトまで様々である．レビュー議事録の登録件数は，約20,000件に達する．

8.5.2 効果

定量的な効果の算出は難しいが，定性的な効果としては以下の4点が挙げられる．

(1) プロジェクトマネジャが設計の進捗を随時把握できる．
プロジェクトマネジャは，特別な報告を求めなくても，レビューの実施状況と内容を随時把握できる．

(2) 問題を早期に発見しやすい．
テーマごとにレビューの実施回数がわかるため，検討漏れが無いか，十分なレビューを実施しているか，レビューの実施順序に矛盾が無いか，作業が停滞していないかといった点からチェックすることで，潜在的な問題を見つけやすくなった．

(3) 設計内容の是非が判断しやすい．
レビュー議事録から，データ設計・インタフェース設計など設計項目毎に議事内容を把握できるので，チェックがしやすくなった．

(4) 効果的な設計レビューが実施できる．
標準テーマはプロジェクトで自由に編集が可能なため，プロジェクト開始時に標準テーマをベースとしてプロジェクト内でレビューテーマを選定し，レビュー計画を立てることで，効果的な設計レビューが可能になった．

本ツールを使いこなせば，以上のような効果がある．本ツールが提供するテー

マごとのレビューの実施回数という情報は，設計の十分性や網羅性を判断するための１つの材料に過ぎない．ツールが示す情報から，プロジェクトマネジャが行動を起こしてはじめて，本ツールは価値あるものとなる．つまり，プロジェクトマネジャが検討の不十分な点，問題が未検討と思われる点にいち早く気づき，対策を打つ，あるいはプロジェクトリーダから進捗状況をヒアリングして，設計状況を把握することが必要である．

本ツールの上記の複合効果により，プロジェクトマネジャは早期に設計の弱点を見つけ出すことができ，設計品質を向上させるのに役立っている．今後機会があれば，本システムの適用による品質向上のデータを分析し，報告したい．

8.6 まとめ

ソフトウェア品質向上のための取り組みとして，レビューの議事録を検索・分析し，レビューを定性的，定量的に評価するツールを開発した．ツールには，レビュー議事録から特定のテーマやキーワードに関する議事内容を検索する機能をもたせ，設計の検討経緯が把握できるようにした．また，レビューの実施状況を分析する機能をもたせ，定量的に把握できるようにした．

さらに，レビュー議事録の分析結果は，進捗状況や設計の品質問題の存在を推測するためのデータとしても活用でき，品質管理はもちろん，プロジェクト管理の観点からも利用可能であることを述べた．

従来の開発では管理されていなかった設計レビューの議論を管理することにより，設計品質の向上を図り，開発するソフトウェアの品質向上に寄与すると考えている．

第９章　プロジェクトナビゲーションツールによる管理精度の向上

9.1　はじめに

業務ソフトウェアの開発では，大規模化，高機能化，短納期化のニーズが高まってきている．これを達成するために，プロジェクト管理の重要性も急速に高まってきている．プロジェクト管理は，計画をたて，その計画の進捗状況をチェックし，問題があれば対策をとるという，Plan-Do-See を繰り返していくことであるが，特に業務ソフトウェアの開発では，顧客側からの機能追加や仕様変更の要求が，非常に多く出ることがあり，計画の変更が頻繁に起きやすい．いずれにしても，プロジェクト管理では，常に計画を明確にしてプロジェクト内に徹底し，正確な進捗状況を効率よく，かつタイムリーに把握し，適切な対策を即時に取るということが重要である．

日立製作所情報グループ生産技術本部では，プロナビと呼ぶ，開発に必要なすべての情報をディジタル化し，管理者，開発者が必要に応じて，自分の PC から，それらの情報を参照したり，再利用できるディジタルエンジニアリングシステムを開発した．プロナビは，プロジェクトの全員に，仮想的なプロジェクトルームを提供するが，これにより，主として以下の 3 点を実現している．

- ●各 Work で実施すべき項目の予定，実績，担当者，状況の記述とそれらの Work 単位の表示(To-Do 機能)
- ●各 Work での成果物の明確化と Work 単位の表示
- ●各 Work で利用する資料の明確化と Work 単位の表示およびリンクによるアクセス(To-See 機能)

開発者は，プロナビを利用することで生産性の向上を図ることができる一方，管理者はプロジェクト管理のためにプロナビを利用することで，プロジェクトの正確な進捗状況を効率よく，かつタイムリーに取得できるため，管理精度の向上を図ることが可能になる．

本章では，まず，プロナビの機能を説明し，プロジェクト管理上，必要な情報をプロナビにより容易に取得できることを説明する．

9.2 プロナビの概要

プロナビは，開発作業者とプロジェクト管理者の双方の作業効率向上を目的としたディジタルエンジニアリングシステムであり，次の３点の大きな狙いをもって開発した．

- ●開発・管理 WBS（Work Breakdown Structure）の標準設定・明示・ナビゲーションによる作業の定型化・効率化
- ●成果物の一元管理と再利用（設計，管理の再利用）
- ●Web/グループウェア技術の利用による分散開発での設計/管理の高効率化

（１）開発・管理 WBS の標準設定・明示・ナビゲーションによる作業の定型化・効率化

この実現がプロナビの最も大きな特長である．

プロジェクトの進行では，まず計画を立てるが，その計画立案では，ソフトウェアの開発プロセスを設計者や管理者の行うべき仕事(Work)に分解し，各 Work での成果物，担当者，期限を明確にする(To-Do)．これらの Work は WBS という順序関係をもった構造体になるが，特に開発者用の Work の構造体を開発 WBS，管理者用の Work の構造体を管理 WBS と呼ぶことにする．次に，この計画に従った遂行であるが，各 Work の作業では，必要な資料を参照しながら進めていく．この参照する資料の標準セットを示したり，簡単に PC 上からアクセスできると効率も良いし，作業の定型化もできる(To-See)．さらに，Work の作業状況を入力したり，管理者は，それをチェックする．

プロナビでは，これらをディジタル化し，管理者，開発者の PC に表示できるようにした．

すなわち，プロナビは，こうしたソフトウェア開発プロセスを，設計者，管理者の行うべき仕事(Work)の WBS をもとに，そこでの成果物，作成者，作成期限，参照すべきドキュメント等を決めたものを，統合的にユーザに見せたり，情報を格納したり，活用する環境を提供する．

こうした WBS は，対象とするシステム形態や業務分野ごとに標準化することが作業を定型化する上で必要であるが，一方，プロジェクトごとに，不要な作業があったり，追加の作業がでてくる．従って，標準化された WBS のプロジェクトごとのカスタマイズが必要である．このカスタマイズが簡単にできるように，WBS の標準パターンを雛型としてユーザに提供する提供することとした．プロジェクトではこの雛型を，適用開始時に必要に応じてカスタマイズして使用する．これにより，WBS の定義にかかる作業の効率化や，必須作業の漏れによる品質低下を防止することができる．

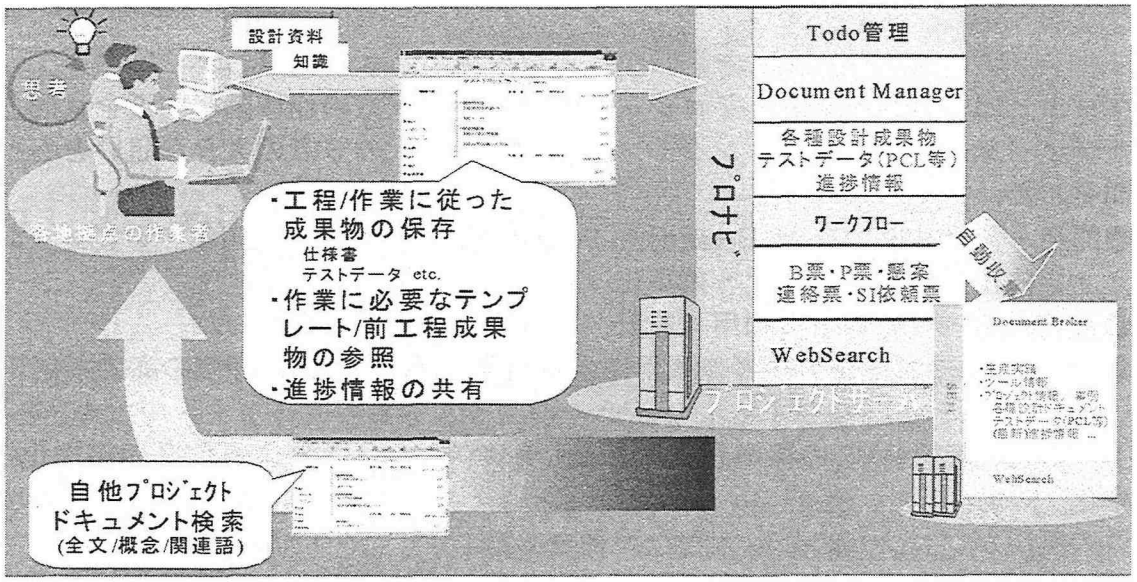


図９．１　成果物の一元管理と再利用

（２）成果物の一元管理と再利用

CASE ツールはプログラムの再利用により生産性を向上させるものであるが，プロナビでは設計･管理の再利用により生産性を向上させることを図っている(図９．１参照)．この実現のために，プロナビのもとで作成し，審査･承認された設計ドキュメント，管理ドキュメントは，全て一元管理し，誰でもが検索し再利用可能にしている．

開発者は設計作業において，他の類似するプロジェクトの成果物を検索して見つけ出し，再利用する．そこで完成した新たな成果物を，プロナビに格納する．この成果物（ドキュメント）は，別のプロジェクトの開発者が参照，再利用したり，同一プロジェクトでの機能拡張作業などの際に，再利用される．

（３）Web/グループウェア技術の利用による分散開発での設計/管理の高効率化

分散している拠点をもつプロジェクトでは，物理的に１つの場所に集まっているプロジェクトに比べて，設計や管理作業の効率が低下する．例えば，ある拠点で作成された設計書を別の拠点にいる人がすぐに見ることができない．この設計書を，他の拠点に届ける場合でも，何かの手違いで届かなかったり遅れたりすることが多い．また，工程の進捗状況，品質状況，懸案事項等，プロジェクト内全員で共有すべき情報が，拠点間で共用できない．こうした問題を回避するためには，開発者，管理者ともに，相当の負荷がかかることになる．しかし，インターネットやグループウェアの普及した現在では，このような課題の解決は非常に簡単であり，プロナビでもこれらの技術を採用することで，分散拠点開発･管理の効率化を図っている．

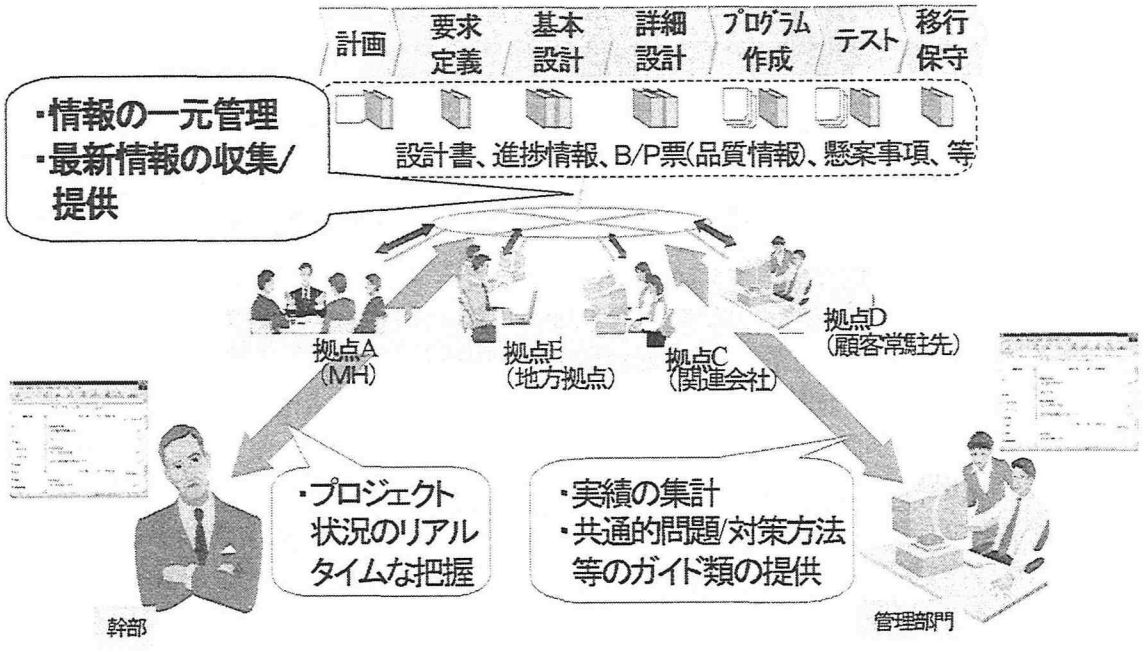


図９．２　分散開発での設計/管理の高効率化

プロナビでは，各開発拠点をネットワークで接続し，情報を格納しているデータベースを一箇所に集約した．また，グループウェアの機能を用いて，送付・開封確認などを容易に行えるようにした．さらに，また，プロナビの各機能を Web ベースで利用できるようにし，開発者や管理者の PC に特別なソフトウェアの組み込みを不要にした（図９．２　参照）．

9.3 プロナビによるプロジェクト管理

前節でプロナビの概要を述べたが，9.3 では実際の開発業務のなかでプロジェクト管理を行う際に，プロナビをどう活用するのかを述べる．

9.3.1 プロジェクト管理と課題

プロナビの活用方法を述べる前に，業務ソフト開発におけるプロジェクト管理では何をすべきか，また，必要な情報収集にあたっての課題を以下に整理する．

（１）　業務ソフトウェア開発におけるプロジェクト管理

プロジェクトは予め決められたコスト，期間内で所定の機能，品質を満たすソフトウェアや情報システムの開発を行わねばならない．これが実現できるように，

プロジェクトを推進していくのがプロジェクト管理である．業務ソフトウェア開発では，進捗，コスト，開発規模，品質，懸案事項の状況について，正確，かつタイムリーに把握することがプロジェクト管理遂行上，特に重要である．

（2） プロジェクト管理遂行上の課題

業務ソフトウェア開発では，次の４つの理由により，プロジェクト管理をする上で重要な情報を，正確かつタイムリーに収集することが難しい．

(a) プロジェクトの大規模化・グローバル化により，開発拠点が分散されていることが多いためである．こうした状況では，プロジェクト管理者は，数多くの拠点に対し，情報の提供を求め，収集・吟味し，集約して判断しなければならない．この情報の提出と収集に，プロジェクト管理者と開発作業者双方に，分散ゆえの負荷がかかる．

(b) PC 上の開発環境の高度化により，ソフトウェア開発そのものが，ある程度まで各作業者個人の中に閉じており，プロジェクトという意識が希薄な状況にあることである．このため何か問題があった場合でも，プロジェクト内の管理者に見えてこないといったことが起こり得る．

(c) 業務ソフトウェアのプロジェクトは，開発が終了するとプロジェクトが解散されるので，繰り返しが行われず，その結果，技術上，管理上のノウハウが蓄積されない．そのために，開発者による種々の管理情報の入力が滞りがちになる．

(d) プロジェクト管理のための情報の収集には，開発作業者の「管理のための情報を提出する」作業，これをまとめるサブリーダクラスの管理者の「集計する」作業が必要であり，負荷が大きい．

したがって，開発作業者にできるだけ負荷をかけずに，最新の正確な情報を収集するための仕掛けが必要であり，これをプロナビで解決した．

9.3.2 プロナビのプロジェクト管理支援機能

（1）機能一覧

プロナビのもっているプロジェクト管理支援機能を，表９.１に示す．プロジェクトワークスペースは，プロジェクトの開発者，管理者に仮想的な作業環境を提供しており，プロナビの大きな特長の１つである．以下，プロジェクトワークスペース機能による進捗管理について述べる．

表９．１ プロナビの提供する主な機能

項番	機能名		概要
1	プロジェクトワークスペース	プロジェクト構成管理	・プロジェクトの親子関係の構成，所属メンバの管理
		WBS 管理	・各ワークの担当者，期限，状態の管理(To-Do) ・当該作業で参照，再利用する資料の明示，リンクによるアクセス等の作業ナビゲーション(To-See) ・標準 WBS の提供
		作業環境の提供	・プロジェクト毎のワーク単位の作成物の履歴つき保管 ・To-Do, To-See とを統合した View の提供
2	日程管理	－	・アロー図の一元管理
3	懸案管理	－	・懸案の発生～解決管理の半自動化
4	不良管理	－	・不良の発生～解決管理の半自動化 ・発生状況等の分析出力
5	仕様変更管理	－	・仕様変更の発生～解決・保留・ドロップの管理の半自動化
6	事例検索	－	・他プロジェクトの設計・管理ドキュメントの検索

（２） プロジェクトワークスペース機能概要

プロジェクトワークスペースは，プロジェクトで設定した WBS に関するあらゆる情報を統合的に管理する環境を提供する．プロジェクトにおけるあらゆる作業は，WBS という構造をもった Work の集合体から成る．プロジェクトワークスペースの環境から，特定の Work をクリックすると，「To-Do」で，この Work で行うべき作業の作成成果物，担当者，期限，マイルストン，現状況が表示される．また，「To-See」で，この Work を行うときに参照すべき資料が表示される．

通常，管理者は開発用の Work の To-Do を参照すれば，プロジェクトの進捗状況の把握ができる．しかし，プロジェクトが巨大な場合には，Work の中の項目が膨大になり，効率が悪い．そのために，管理用の Work を設定できるようにした．管理用 Work の To-Do にレビューや顧客調整等のマイルストンを設定すればよい．

図９．３に，プロジェクトワークスペース機能の画面と概要を示す．

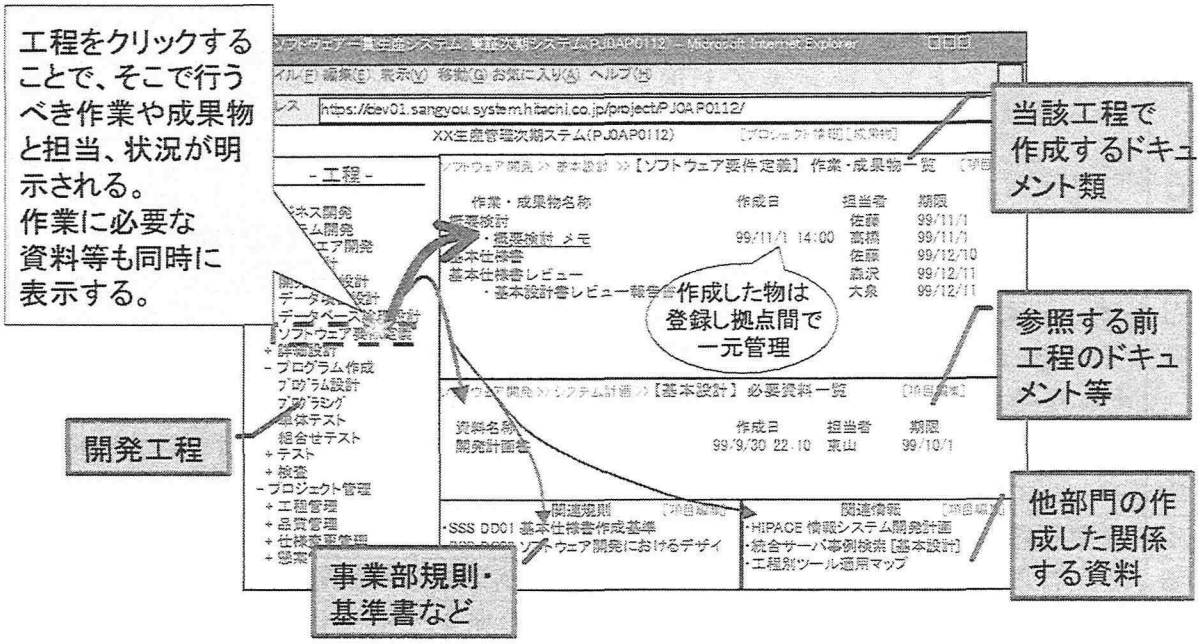


図　9．3　プロジェクトワークスペース機能概要

（3）プロジェクト開始時のワークスペースへの初期設定

プロジェクトの開始時に，プロジェクト管理者は，提供された雛型から，そのプロジェクト用に作業の追加や名称変更などのWBSのカスタマイズを行う．次に，そのＷＢＳ内の各作業に対し，担当者と期限を設定する．この設定は，プロジェ

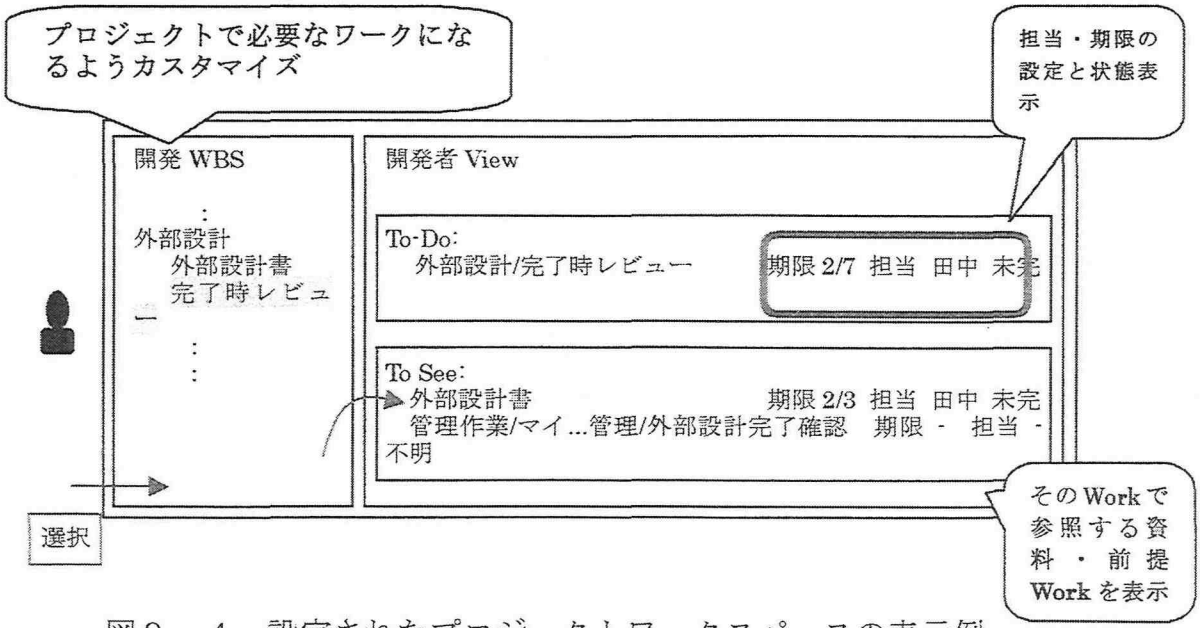


図９．４　設定されたプロジェクトワークスペースの表示例

クト管理者だけではなく，開発作業者のリーダやサブリーダが行うこともできるようになっている．

こうして設定されたプロジェクトワークスペースの表示例を図９．４に示す．

（４）開発 WBS と管理 WBS の関連付け

開発 WBS と管理 WBS の２つが存在する場合，相互の関連付けが必要となる．例えば，開発 WBS に「xx 仕様書レビュー」と設定されていれば，実施状況を管理 WBS からチェックできるようにしたい．この逆の場合もありえる．このように相互に関連する作業については，作業の主体者の WBS の To-Do に作業を記述し，主体者でない方の WBS の To-See から状況を見ることができるように関連付けをした．

（５）プロジェクト遂行時の進捗状況の記録と把握

プロジェクトワークスペース上で，開発者およびプロジェクト管理者は作業を進めていく．開発者は，自分の担当作業を確認し，その作業を遂行するとともに，遂行状況をプロジェクトワークスペースに記録する．特に，ドキュメントについてはワークフローによる審査･承認も状況が自動的に記録されるようになっている．プロジェクト管理者は，開発 WBS の中もしくは管理 WBS の中をチェックすることで，進捗状況の把握ができる．

具体例として，外部仕様設計を行う時を考えてみる．

業務ソフトウェア開発プロジェクトでは，取扱データ，帳票，画面表示内容，操作，性能等の仕様を記述した設計ドキュメント（仕様書）を作成する．次に，この仕様書に記述してあるに内容について，実現可能性，コスト，顧客の要求仕様充足度などの観点から，レビューを実施する．これらの一連の作業は，プロジェクトワークスペース上に開発 WBS としてあらかじめ展開されており，開発者は，対応する作業で作成した設計ドキュメント等の成果物を登録するとともに，その作業がどこまで進んでいるかを記録する(図９．５)．

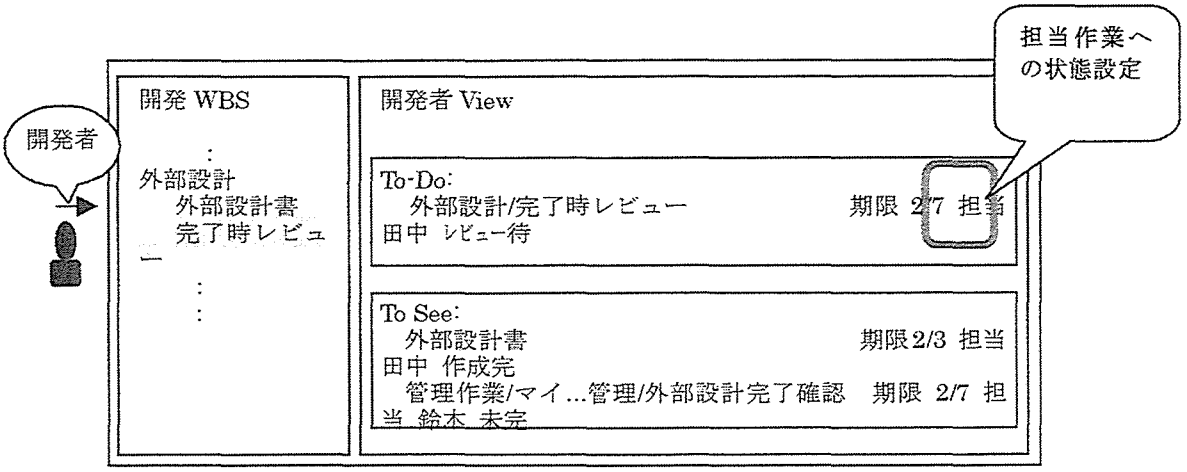


図９．５　プロジェクト遂行時の進捗状況の記録と把握

プロジェクト管理者は，プロジェクトワークスペースの管理者の View で，プロ

ジェクト管理者の担当となっているワーク部分を見ればよい．それを図９．６ で示している．

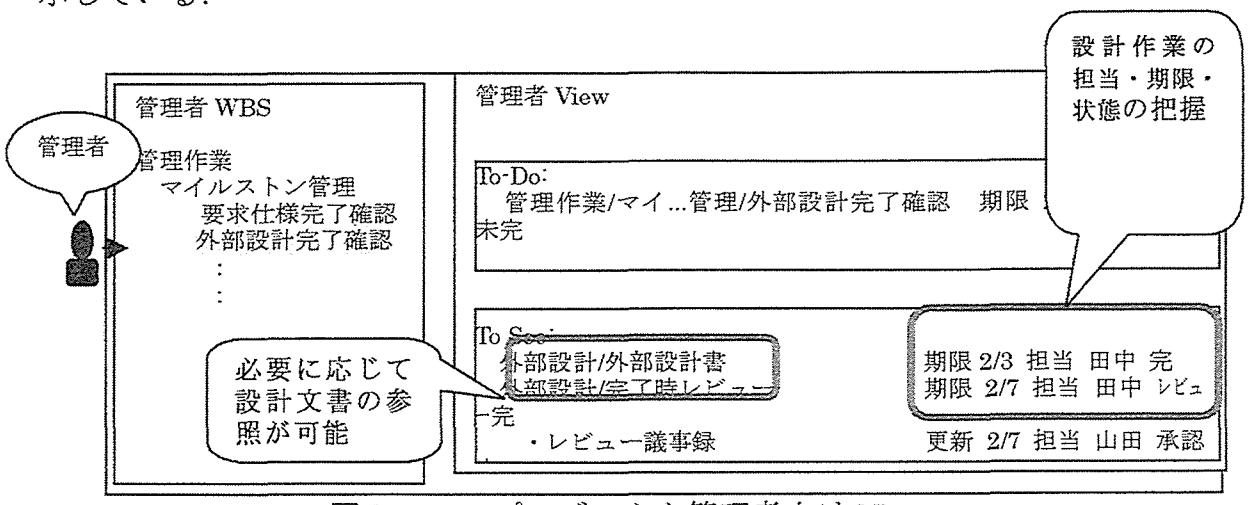


図９．６　プロジェクト管理者向け View

プロジェクト管理者は，この画面から，「マイルストン管理」の中の「外部仕様完了確認」作業が自分の担当であること，それを完了させるために必要な開発の作業「完了時レビュー」の作業の担当・期限・状態を把握することができる．また，これに関係する成果物を確認したければ，その設計文書等を，同じ画面上から取り出して参照することができる．

これにより，プロジェクト管理者は，プロジェクトワークスペースを通じて，進捗の情報を把握することが可能である．

9.4　評価と今後の計画

9.4.1　適用状況

プロナビは，2000 年 3 月に本格適用を開始し，現在までの間に，300 を超える業務ソフトウェア開発プロジェクトに適用している．これらのプロジェクトは，銀行・製造業・流通業・官公庁などのユーザ業務用のシステムを開発する大規模プロジェクトから，社内の事務や開発支援のツールを開発する小規模プロジェクトまで様々である．

9.4.2　効果

プロナビの，プロジェクト管理における定量的な効果の算出は難しいが，定性的な効果としては以下の５点が挙げられる．

（1）プロジェクト状況把握の効率向上が図れる．
プロジェクト管理者は，開発者から特別な報告がなくても，各作業の担当者，進捗状況，期限の正確でタイムリーな把握が可能となった．開発者から見ても，管理資料作成に手間取られず，開発の作業に注力できるようになったこともあげられる．さらに，こうした情報は分散拠点間でも一元管理されて，収集・送付・フォローといった手間を大きく減らすことができている．

（2）現物確認がすぐにできる．
単に開発者からの報告だけでなく，設計ドキュメントやレビュー報告書の内容をプロナビから直接，かつ即座にチェックすることができる．

（3）雛型 WBS 活用による計画作業の効率向上が図れる．
プロナビから提供される WBS の雛型をカスタマイズすればよいので，プロジェクト開始時に具体的な計画を容易に作成できるようになった．また，作業項目の設定洩れが防止でき，精度の高いプロジェクト計画を設定できるようになった．

（4）課題の顕在化を容易にできる．
これまで，顕在化しづらかったプロジェクトの課題を，プロナビが提供する懸案管理，仕様変更管理，不良管理等の現物を見ることで，容易にしかも早期に把握できるようになった．

（5）プロナビは開発そのものに有効であるために，開発者が積極的に利用している．そのために，管理用のデータを入力することに対するアレルギーが非常に少ない．

9.5 まとめ

プロジェクト管理者および開発者の効率向上を目指したディジタルエンジニアリングシステム「プロナビ」を開発し，適用をしてきた．プロナビの大きな特長は WBS という観点からの，作業項目の予実績把握，作業ナビゲーション，成果ドキュメント管理とグループウェア技術による情報共有である．プロナビを利用することで，プロジェクト管理者は，開発者が入力した実績情報や，成果物である設計ドキュメント，レビュー報告書，懸案事項一覧，不良発生状況，仕様変更状況等の現物をチェックすることができ，本来の目的であったプロジェクト状況の正確でタイムリーな把握を容易にできるようになり，管理精度の向上に役立っている．

第10章　ソフトウェア品質評価精度の向上と効率的なフィードバック手法

10.1　はじめに[30]

情報システムが大規模化,複雑化している中において,品質管理の重要性は益々強く要望されている．効果的な品質管理を行うためには，品質データをいかに迅速に的確に収集・分析し，開発工程にフィードバックさせることができるかが重要となってくる．

日立製作所情報グループの品質保証統括センタでは，近年，ソフトウェア開発が短納期化，低コスト化してきていることへの対策として，従来からの品質評価手法を応用し，品質データの評価精度を向上させ，効率的にフィードバックさせるための手法を確立させた．

当手法は不良密度，テスト項目密度，テストカバレージ率等品質指標の相関評価の組合せをパターン化し，品質評価精度の向上を図っている．さらに品質指標の目標値に対する達成度を点数評価する手法により，弱点プログラムを抽出し，効率的な品質向上を図れるようにしている．

これらの手法は，品質データのタイムリーな収集と分析を行うためのしかけと合わせ，上記品質保証統括センタで開発した総合的品質管理ツールであるQE-EXPERT(Quality Evaluation－EXPERT)に取り込み，社内プロジェクトへ適用している．

本章では，品質データの評価精度の向上と効率的なフィードバック手法およびQE-EXPERTをプロジェクトへ適用した際の効果について述べる．

一方，従来からテスト工程途中で残存不良件数を予測するために,「ソフトウェア信頼度成長モデルによる信頼性予測」を実施し，信頼性向上に寄与してきた．これを顧客納入後の稼動段階での信頼性予測に使用したいという願望を抱いてきたが，予測と実際のデータにはギャップがあった．最近になって，稼動後の信頼性予測が実現可能に近づきつつあるので，その状況についてもあわせて述べる．

10.2　近年の品質管理上の問題点

ＰＣでの開発が主流になった現在，メインフレーム系開発も含めほとんどがＰ

C上で品質管理を行うようになり，かつ表計算等のアプリケーションソフトウェアの充実により，個別の専用ツールが無くても誰もが簡単に不良の分析が出来るようになった．このことにより，プロジェクト固有の分析，評価が素早く柔軟に行えるようになるという利点が増えた．

しかしながら，各プロジェクトで独自に分析ツールを作るケースが多くなったがために，逆に以下の問題が発生するケースが目立ってきた．

①プロジェクト間の品質管理スキルのバラツキが顕著になった．
②品質管理のしくみを作り軌道に乗せる作業が後手にまわり，有効な管理ができないプロジェクトが増えてきた．

品質管理に求められるものは，これらの問題点の解決を図ることと同時に，PC上での柔軟で自由度のある品質管理を実現することであった．このため，従来からの品質管理方法に改善を加えた形の総合的品質管理ツール QE-EXPERT を開発することになった．表１０．１に，品質管理上の問題点(課題)を解決すべくQE-EXPERTによって実現した内容を示す．

以下，品質評価精度の向上方法，効率的なフィードバック方法およびQE-EXPERTとプロジェクト管理の流れについて述べる．

10.3 品質評価精度の向上と効率的なフィードバック方法

品質評価精度を向上させると共に，効率的にフィードバックする方法について以下に示す．

10.3.1 品質評価精度の向上方法

（１）相関評価による評価精度の向上

品質データの評価精度向上のために，チェックリスト密度，不良密度およびカバレージ率等の各品質データ単体の目標達成度だけでなく，それぞれの指標の相関関係を使って評価を行うことで，評価度を向上させることが出来る．

ここで，本章で使用する一般的ではない用語の説明を行う．

・チェックリスト：本章では，テストケースあるいはテスト項目の意味で使用．
・カバレージ率：テストカバレージ率の意味で，その定義はC1メジャーといわれているもの．すなわち，プログラム内の全分岐数に対する当該テストで実行した分岐数の比率．

表１０．１ 品質管理上の問題点と課題

項番	品質管理上の問題点(課題)	QE-EXPERT 実現内容	備　考
1	プロジェクト間の品質管理スキルのバラツキが顕著.	品質評価方式を確立させるべく,従来からの評価方式に加え「相関評価による評価精度の向上と相関評価パターンによる自動評価」を図った.	１０．３．１「評価精度の向上方法」参照
2	評価効率の向上	上記に加え,効率的なフィードバック方法を図るため,「達成度の点数付けによる弱点機能の抽出」を図った.	１０．３．２「効率的なフィードバック方法」参照
3	品質管理のしくみ構築が遅れ,有効な管理ができない.	品質管理に必要なデータをタイムリーに収集可能なように,「プロジェクト管理全体を支援するツール群」を開発する.	１０．４「QE-EXPERTとプロジェクト管理の流れ」参照
4	ある一定の品質管理レベルを維持し,柔軟で自由度のある評価を可能とする.	プロジェクト固有の評価が可能なようにデータの公開とツールのカスタマイズが可能となるようにした.	同　上

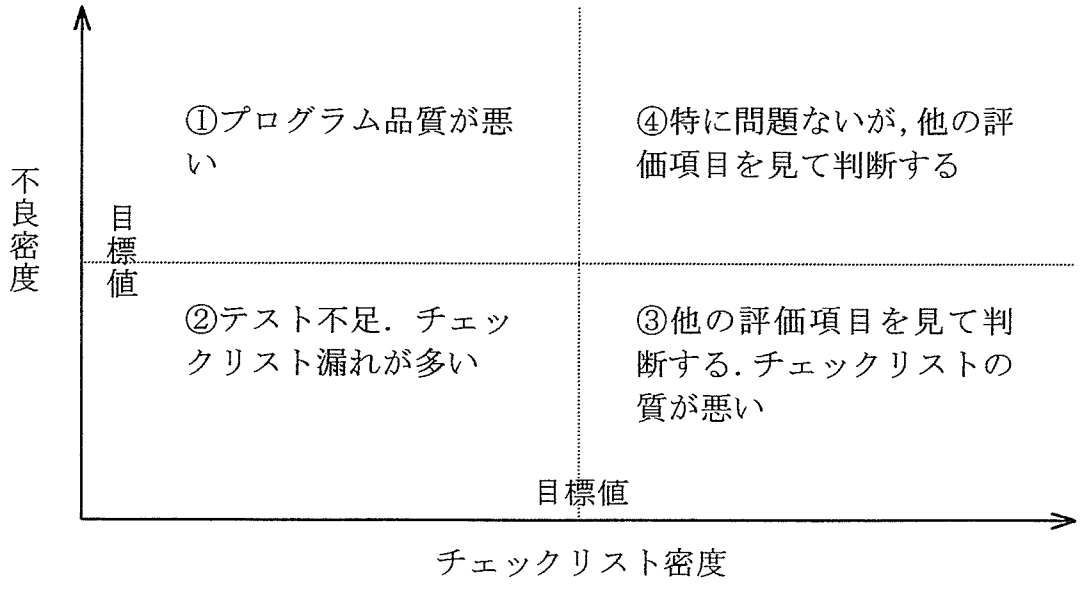


図 10.1　バグ摘出十分性評価

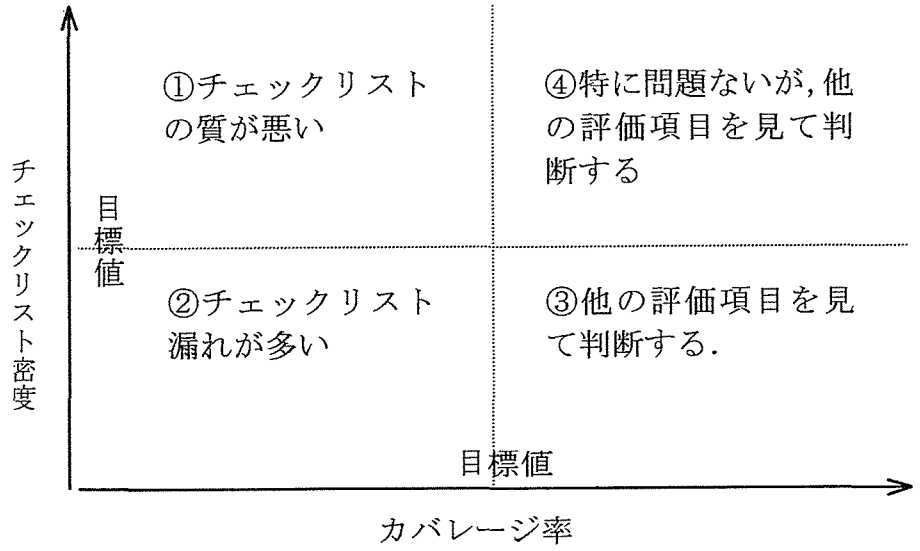


図 10.2　チェックリスト十分性評価(1)

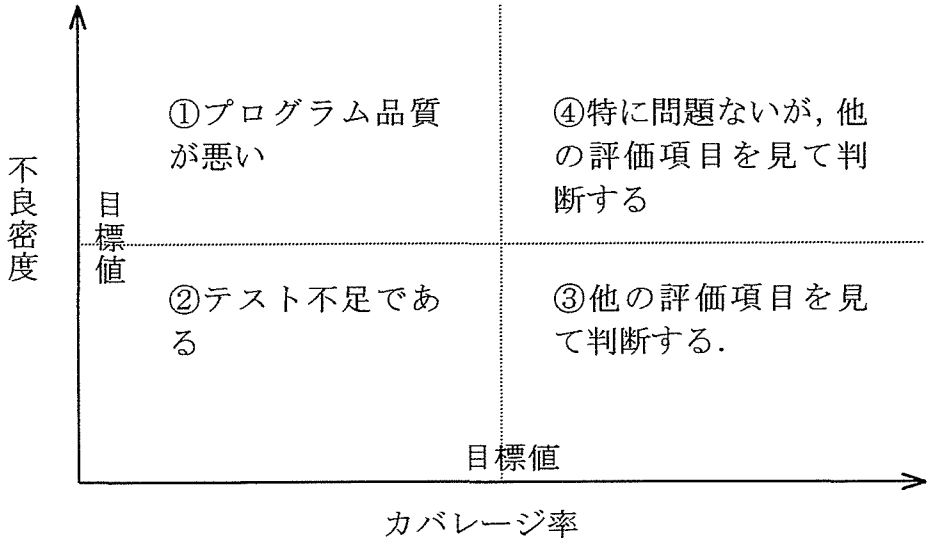


図 10.3　チェックリスト十分性評価(2)

図１０．１に，チェックリスト密度と不良密度の相関評価内容を示す．これから明らかに分かることは，チェックリスト密度が目標値より小さく不良密度が目標値より大きい場合，プログラム品質が悪いということである．

図１０．２に，カバレージ率とチェックリスト密度の相関評価内容を示す．これから明らかに分かることは，チェックリスト密度が目標値より大きくカバレージ率が目標値より小さい場合，チェックリストの質が悪いということである．

図１０．３にカバレージ率と不良密度の相関評価内容を示す．ここで不良密度の達成度だけでは不良摘出の十分性が判断しにくいが，客観的なテスト十分性の指標であるカバレージ率との相関を取ることで本当に不良の摘出が十分かの信憑性が増す．すなわち，カバレージ率が目標値より小さいにもかかわらず，不良密度が目標値より大きい場合，プログラム品質が悪く，カバレージ率が目標値より小さく不良密度が目標値より小さい場合は，チェックリストの質が悪くテストが不十分の可能性が高いと考えられる．

不良密度，チェックリスト密度，カバレージ率という３つの指標の相関評価を調べることにより，プログラム品質が悪いのか，テストの質を左右するチェックリストの質が悪くテストが不十分なのかというような，品質評価の重要な判断材料が得られることがわかる．

（２）相関評価のパターン化

（１）で示した相関評価の考え方を基本として，より評価精度を高めるため，相関評価の指標としてチェックリスト密度，不良密度，カバレージ率の３指標の目標値達成・未達成の組合せをパターン化した．対象プログラムがどのパターンに属するかによって，品質状況を自動的に見極めることができる（図１０.４参照）．

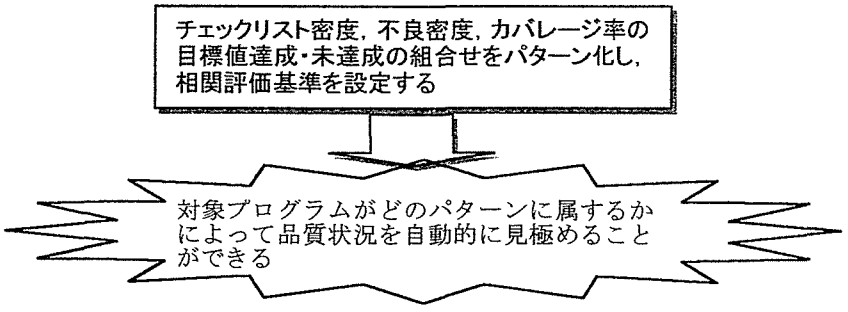


図 10.4 相関評価のパターン化

表10.2にカバレージ未取得時相関評価パターン表を，また表10.3にカバレージ取得時のパターン表を示す．ここで，不良密度については不良摘出目標値を達成しても作り込み品質が悪い場合，不良が残存している可能性が高いため，不良密度がある値（通常は目標値の２倍）を超えた場合，アラームを出すような評価方法とした．

表 10.2 カバレージ未取得時相関評価パターン表

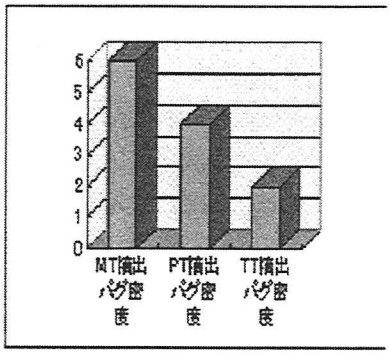
パターン	目標値達成度			品質評価
	カバレージ率	チェックリスト密度	不良密度	
1	なし	未達	未達	テスト不足・チェックリスト件数不足
2			達成	チェックリスト件数不足・作り込み品質悪し
3			n倍以上	チェックリスト件数不足・作り込み品質悪し
4		達成	未達	作り込み品質良い またはチェックリスト内容不十分
5			達成	不良摘出完了
6			n倍以上	不良摘出完了（作り込み品質悪し）

表 10.3 カバレージ取得時相関評価パターン表

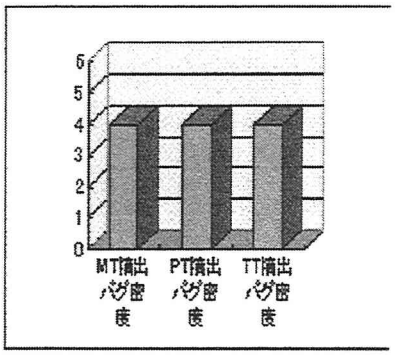
パターン	目標値達成度			品質評価
	カバレージ率	チェックリスト密度	不良密度	
1	未達	未達	未達	テスト不足・チェックリスト件数不足
2			達成	チェックリスト件数不足・作り込み品質悪し
3			n倍以上	チェックリスト件数不足・作り込み品質悪し
4		達成	未達	チェックリスト内容不十分
5			達成	チェックリスト内容不十分・作り込み品質悪し
6			n倍以上	チェックリスト内容不十分・作り込み品質悪し
7	達成	未達	未達	不良摘出完了（作り込み品質良い） または，チェックリスト内容不十分（分岐組合せ等）
8			達成	不良摘出完了 または，チェックリスト内容不十分（分岐組合せ等）
9			n倍以上	不良摘出完了（不良多発） チェックリスト内容不十分（分岐組合せ等）
10		達成	未達	不良摘出完了（作り込み品質良い） またはチェックリスト内容不十分（分岐組合せ等）
11			達成	不良摘出完了
12			n倍以上	不良摘出完了（作り込み品質悪し）

（３）工程間相関評価による評価精度の向上

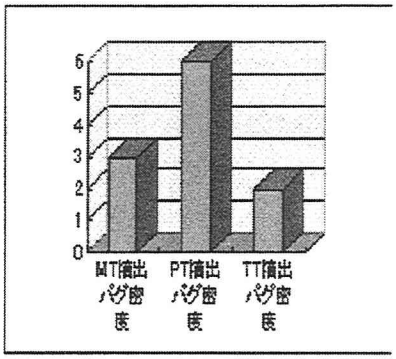
前工程の不良摘出十分性は，次工程以降の不良発生状況から判断できる．次工程ではどの工程で摘出すべき不良であったかの分析を行い，前工程で摘出すべき不良が発生している場合は見直しを図る．図10.5に，不良推移パターンと不良傾向を示す．



①単調減少形
MT,PT,TT と工程推移に連れて減少している場合，品質は安定傾向にあるといえる．



②単調非減少形
工程推移してもほぼ同等の不良が摘出されている場合品質は悪い可能性が高い．前工程で摘出すべき不良を次工程に残していると考えられる．



③増大から減少に転ずる形
PT 工程において大部分の不良を摘出できたと考えられる．

*ＭＴ／単体テスト，ＰＴ／組合せテスト，ＴＴ／顧客テスト

図 10.5 工程別不良推移

10.3.2 効率的なフィードバック方法 ～評価点数の導入～

相関評価のパターンにより，概ねの評価を行うことができる．しかし，相関評価パターンだけでは目標値との差が大きいものでも小さいものでも，等しく目標未達となり同じパターンとなる．例えば，カバレージ率目標値８０％に対して，実績値が１０％のものと，７５％のものが同じパターンに属することになる．このように，相関評価のパターン化により指標を単体で見るより評価精度が向上するが，パターンの中での優先度が見えない．そこで，ＱＥ－ＥＸＰＥＲＴでは各評価要素の目標値に対する達成度を点付けすることとした（図１０．６参照）．

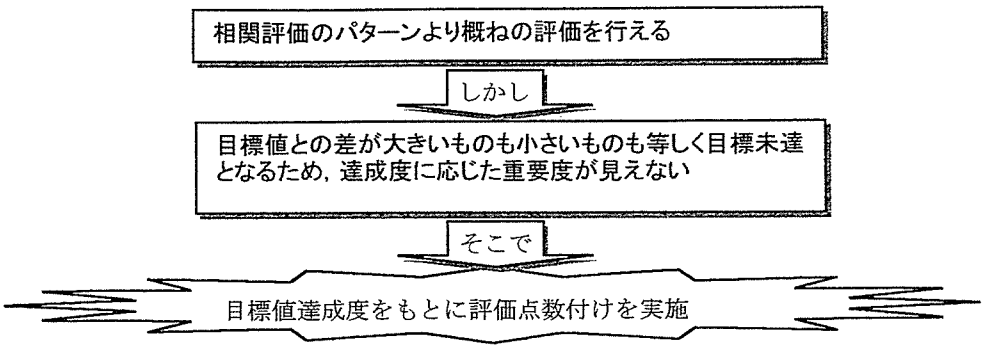


図 10.6 評価点数の導入

（１） 評価点数算出方法

目標値を達成していれば（実績値が目標値以上であれば）１点とし，目標値未達であれば，実績値を目標値で割った値（０以上１未満）を点数とする．

① 目標値≦実績値（目標達成の場合）

評価点数加算値＝１

＊不良密度がｎ倍以上の考慮

不良密度がｎ倍以上でも評価ランクは１点とする．これは，品質向上すればするほど，評価ランクが下がるようなことのないよう考慮した為である．

ただし，出力帳票には不良密度がｎ倍以上であるものにはアラームを記載する．

② 目標値＞実績値（目標未達の場合）

評価点数加算値＝実績値／目標値

チェックリスト密度，カバレージ率，不良摘出密度の実績値からそれぞれ点数をつけ，３つの点数を合計する．

（２） 評価点数算出方法（事例）

表１０．４に評価事例を示す．例１のプログラムは，チェックリスト密度，カバレージ率，不良摘出率のどれも目標値を超えており，評価ランクはすべて1点で合計3点となる．ただし，不良摘出率は２倍以上である． 例２，例３，および例４はチェックリスト密度，カバレージ率，不良摘出率のどれも目標値を下回っており，相関評価パターンは同じとなるが，その目標値達成度より，例２は合計2.2点，例３は合計1.2点，例４は合計0点と優先度が付けられる．

表10.4　評価点数算出方法（事例）

No	ﾓｼﾞｭｰﾙID	定量的品質評価結果												相関評価ﾊﾟﾀｰﾝ文言
		PCL密度			ｶﾊﾞﾚｰｼﾞ率			不良摘出率				総合	ﾊﾟﾀｰﾝ	
		目標	実績	得点	目標	実績	得点	目標	実績	警告	得点	得点	No	
1	機能1	100	120.00	1.00	80	100.00	1.00	10	20.00	★	1.00	3.00	12	不良摘出完了(作り込み品質悪し)
2	機能2	100	80.00	0.80	80	50.00	0.60	10	8.00		0.80	2.20	1	ﾃｽﾄ不足･ﾁｪｯｸﾘｽﾄ件数不足
3	機能3	100	40.00	0.40	80	20.00	0.30	10	5.00		0.50	1.20	1	ﾃｽﾄ不足･ﾁｪｯｸﾘｽﾄ件数不足
4	機能4	100	0.00	0.00	80	0.00	0.00	10	0.00		0.00	0.00	1	ﾃｽﾄ不足･ﾁｪｯｸﾘｽﾄ件数不足

10.4　総合的品質管理ツールＱＥ－ＥＸＰＥＲＴへの取り込み

これまで説明した定量評価手法は,社内の総合的品質管理ツールであるＱＥ－ＥＸＰＥＲＴに取り込んだ．Ｂ／Ｐ票（バグ一覧票／不良対策票）管理ツールのインタフェースより，タイムリーな品質データの収集が可能となる．ＱＥ－ＥＸＰＥＲＴツール群は，Ｂ／Ｐ票管理，品質管理，進捗管理等目的別に使用でき，それぞれインタフェースを有しているので，効率的な運用が可能である（図１０．７）参照.

10.5　ＱＥ－ＥＸＰＥＲＴの適用方法

机上デバック工程までは，プロジェクト管理チームの要員が品質計画とＱＥ－ＥＸＰＥＲＴ適用および各種規準を作成する．机上デバック工程以降は，開発者が各工程の中間，最終で評価を実施し，プロジェクト管理チームは評価支援を行う.

10.6　ＱＥ－ＥＸＰＥＲＴの効果

現在のところ，ＱＥ－ＥＸＰＥＲＴを運用しているプロジェクトは２８プロジェクトである．そのうち効果を計測したプロジェクト事例（Ａプロジェクト）について述べる．Ａプロジェクトではフェーズ２よりＱＥ－ＥＸＰＥＲＴを運用したため，フェーズ１との比較について以下に述べる.

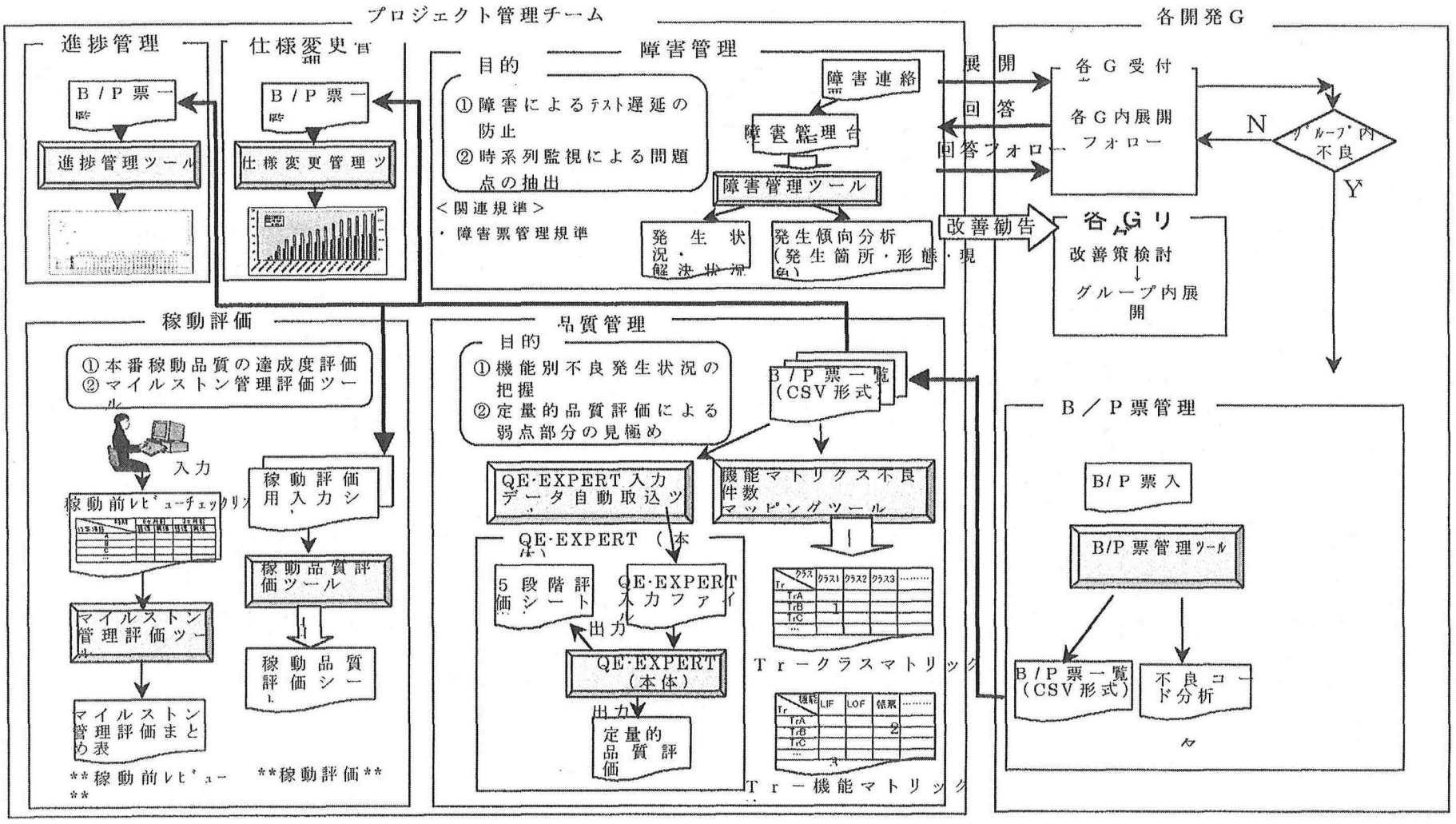


図 10.7　ＱＥ－ＥＸＰＥＲＴ

10.6.1 効率上の効果

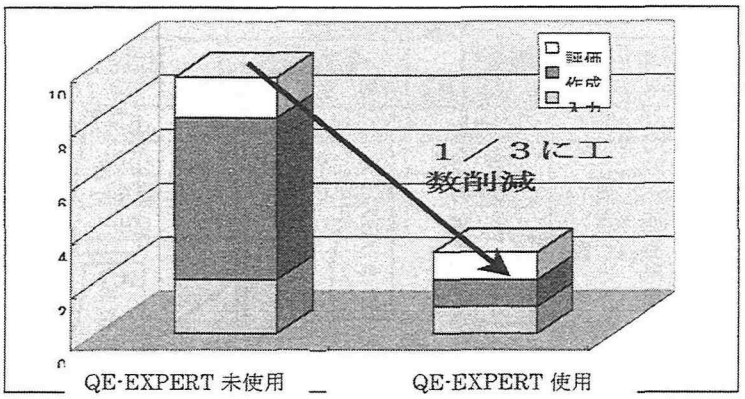
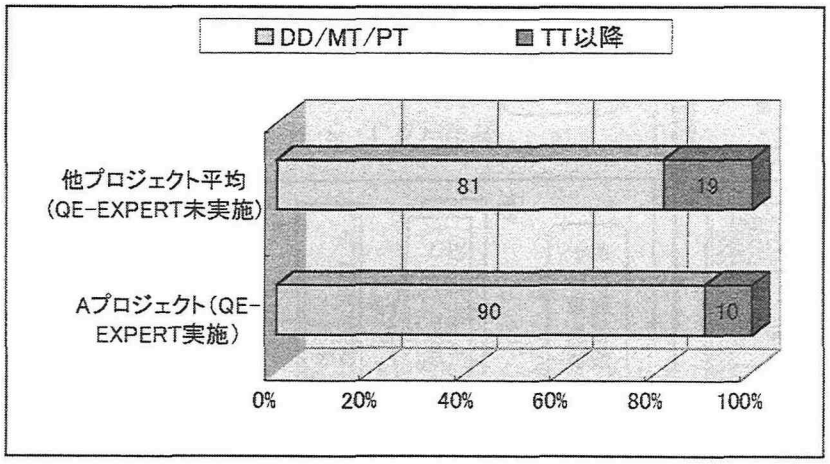


図 10.8 効率上の効果

一度の評価にかかる工数をＱＥ－ＥＸＰＥＲＴ使用時と未使用時で比較すると，以下の（１）～（３）のような効果が得られ，合計で約１／３の工数削減となった（図１０．８参照）．

（１） Ｂ／Ｐ票管理ツールからの自動取り込みの仕組みによりデータ収集が容易となったため，データ集計工数が１／２となった．
（２） 総合的評価資料を自動作成できるようになったため，評価資料作成工数が１／６となった．
（３） 自動評価機能により，評価工数が２／３となった．



10.6.2 品質上の効果

図 10.9 品質上の効果

Ａプロジェクトと類似の過去の品質データ比較を行った．図１０．９は，全工

程で摘出された不良のうち，ＤＤ／ＭＴ／ＰＴ*という製造工程で摘出された不良と，ＴＴという顧客テスト以降に摘出された不良の比率を示す．経験上，全テスト工程に対する上流テスト工程でバグ摘出割合が高ければ高い程，品質は良好であることが分かっている．Ａプロジェクトでは，製造工程の不良摘出率は開発全体の 90%を摘出している．この値は，ＱＥ－ＥＸＰＥＲＴ未使用の他プロジェクト平均の 81%より高く，上流テスト工程での品質確保が行なわれたと言える．プロジェクト特性によって若干の効果に違いがあるものの，他プロジェクトも同程度の効果が期待できる．

10.7　ソフトウェア信頼度成長モデルによる稼動後の信頼性予測

従来，「ソフトウェア信頼度成長モデルによる信頼性予測」は，テスト工程途中で残存不良件数を予測するために用いられ，信頼性向上に寄与してきた．

（１）稼動後の信頼性予測が困難であった理由

この手法を顧客納入後の稼動段階での信頼性予測に使用したいという願望は以前からあり，適用を試みたが，なかなか予測と実際のデータが合わなかった．その理由は以下の通りである．

- 品質が十分でなく，顧客納入後初期不良が発生する．出荷以前のテスト工程や検査工程での着眼点と，顧客の使用の観点が異なるために，不良発生の状況が異なる．
- 出荷以前の不良の発生状況から，顧客納入後の不良発生状況を予測することは困難であった．

（２）稼動後の信頼性予測が実現可能に近づいた理由

- 出荷時の品質が向上し，顧客納入後初期不良が極めて少なくなった．
- 出荷以前のテスト工程や検査工程での着眼点を顧客使用の観点に近づけることが可能になり不良の発生自体が減少するとともに不良発生状況が類似してきた．

（３）ソフトウェア信頼度成長モデルによる稼動後の信頼性予測事例［31］

テスト工程終了時の残存不良件数，つまり稼動後のソフトウェアにどの程度不良が残存するかの予測精度を，実測の稼動後不良件数と各モデルの予測値の相対誤差から評価する．

（ａ）　モデルの評価方法

---

*ＤＤ：机上デバック，ＭＴ：単体テスト，ＰＴ：組合せテスト，ＴＴ：顧客テスト

使用する実際の観測データは組合せ工程から稼動日までの実績不良件数で，ゴンペルツ曲線モデル，指数形ソフトウェア信頼度成長モデル，遅延S字形ソフトウェア信頼度成長モデル，超幾何分布モデルの４モデル[32]に適用し，稼動後の不良件数を予測した（図 10.10 および図 10.11 参照）．対象システムは１６システムである．すべて金融系システムで，ホスト系システムが９システム，クライアントサーバ系システムが７システムであり，規模は約１００KLOC である．

評価指標は，モデルを用いて予測した稼動後の不良発生件数と実際摘出された実測不良発生件数の相対誤差

$$(\text{相対誤差}) = \frac{|M - m|}{M} \times 100\ (\%),$$

M＝モデルによる総予測不良件数，

m＝稼動後３ヶ月までの総実測不良件数

を用いる．

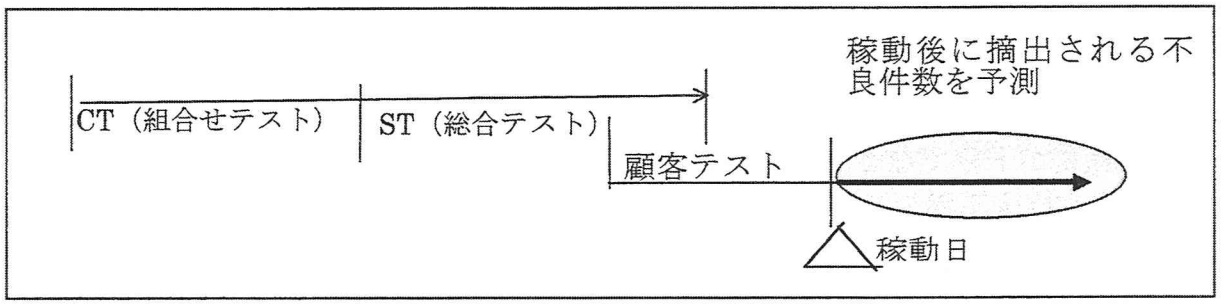


図 10.10 工程の流れと不良件数の予測

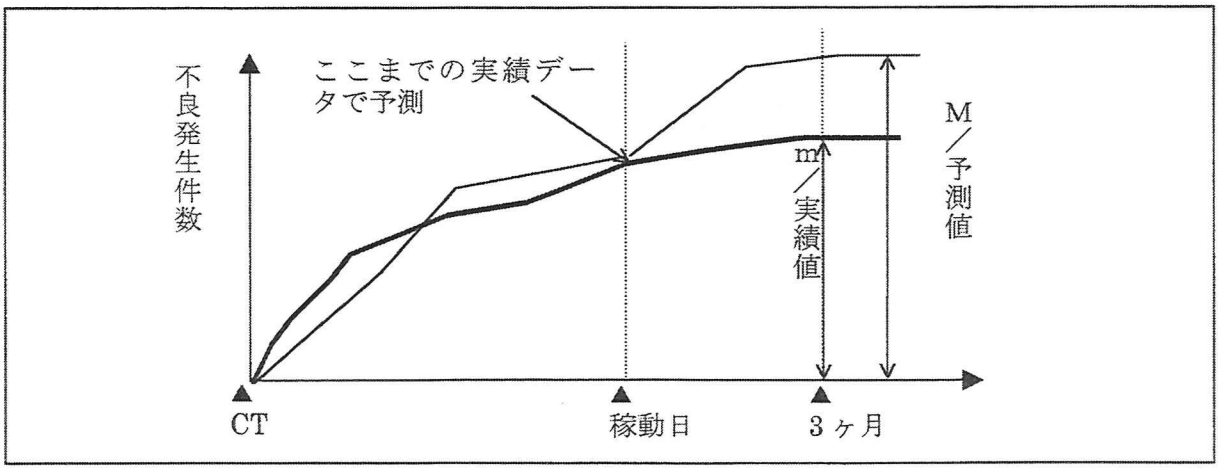


図 10.11　実測値と予測値の誤差

（b）モデルの予測結果と評価

図１０．１２に，各モデルで予測した稼動後の不良件数と実測の稼動後不良件数の相対誤差（１６システム平均）を示す．１６システムの相対誤差の平均は，どのモデルも１７～２６％という結果となった．この結果は，相対誤差の平均としては十分実用的な値であるが，異常値つまり実測の稼動後不良件数が予測より異常に高い場合があるとするとまずいため，その検証が今後の課題である．

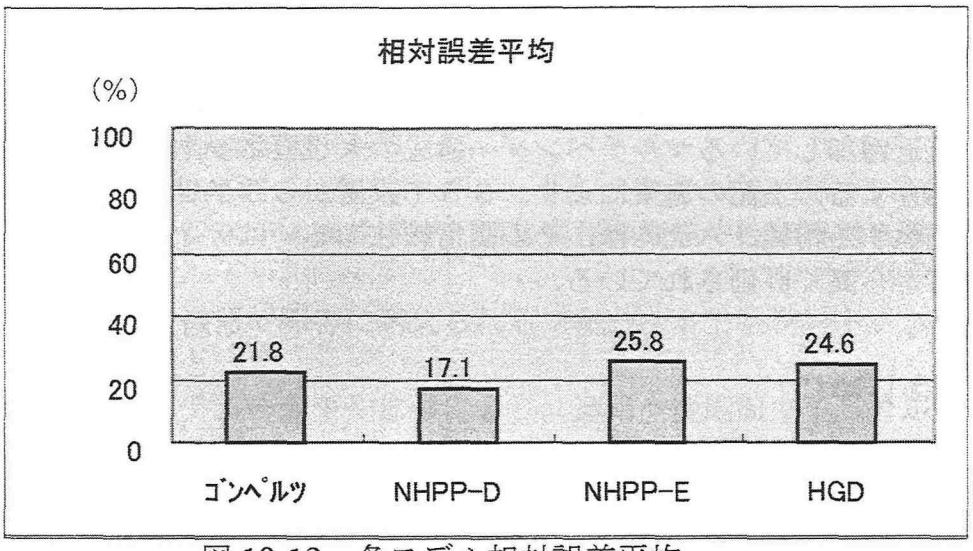


図 10.12　各モデル相対誤差平均

（ＮＨＰＰ－Ｄ(%) 遅延Ｓ字形ソフトウェア信頼度成長モデル，ＮＨＰＰ－Ｅ：指数形ソフトウェア信頼度成長モデル，ＨＧＤ：超幾何分布モデル）

10.8　まとめ

効果的な品質管理を行うため，「評価精度の向上」と「効率的なフィードバック」方法を立案し，QE-EXPERT ツールとして実現した．また，品質データのタイムリーな収集のため，種々のプロジェクト管理を支援するツール群を開発した．

QE-EXPERT を適用することによる効果として，評価工数の低減と早期品質確保が期待できるとの結果が出た．しかしながら，ツールをいくら充実しても使用者による理解不足や適用時期の遅れ，評価結果のフィードバックを怠る等運用方法を誤ると有効な品質管理は出来ない．今後は，QE-EXPERT の適用方法，有効な使い方，および運用方法を含めたドキュメントを充実し，品質管理の質を高めていく必要がある．

また，ソフトウェア信頼度成長モデル［３２］による稼動後の信頼性予測も試みた結果，ある程度期待のもてる結果であった．しかしながら，まだ，出荷前と稼動後の使用環境の差とか，信頼性予測を変動させる因子を予測段階で適切に判断するまでに至っていない．今後さらに検討を重ねていきたい．

第１１章　マルチベンダー構成の情報システム向け システムシミュレーションテスト(SST)

ＳＳＴを実現するために，専用のＳＳＴセンタを日立製作所内に持ち，高負荷や障害を発生させる各種ツールを開発している．日立製作所情報グループの品質保証部門がこれら設備運営およびＳＳＴのテスト技術のノウハウの蓄積・活用を行い，個々のＳＳＴテストは当該顧客担当ＳＥ部門が行う．具体的ＳＳＴ事例として，最近増加しているマルチベンダー構成の大規模ネットワークシステムのＳＳＴを紹介する．上記の施策により，ＳＳＴ設置から四半世紀にわたり，日立製作所が提供する情報システムは，その高信頼性あるいはディペンダビリティに対して顧客から高く評価されている．

11.1　はじめに

日立製作所は１９６０年代から商用コンピュータの開発に着手し，以来，特にオンラインリアルタイムシステム等公共性の高い大規模コンピュータシステムの提供では，高信頼性を追求してきて高い評価を得てきた[1]，[33]，[34]，[35]．こうした日立製作所の高信頼性へのこだわりは，ハードウェアのみならず「ソフトウェア工場方式」に代表される高いソフトウェア品質保証レベルを実現したが，それでも重大障害は皆無にならなかった．結局，障害が起こるのは，出荷前のテスト環境・手段と顧客の運用システムが異なることに起因するためである．そこで，２０年前，ある重大障害を契機にして，経営トップの決断で，以下の対策を講じることにした．この対策とは，顧客先での総合テストや受け入れテストに先立ち，顧客の本番運用に限りなく近いテストを出荷前に社内設備で実現することである．これが本章で取り上げたシステムシミュレーションテスト（System Simulation Test，ＳＳＴと略す）開始のいきさつと，SSTの基本的な考え方である．ＳＳＴ実現のために，大規模な設備投資をして，システムシミュレーションテストの専用設備をもつ計算機センタを構築するとともに，ＳＳＴを効果的に活用するため，製品対応の事業所組織とは別の横断的な専任組織等を作った．

11.2　ＳＳＴの概要[36]

本節では，ソフトウェア製品群を主体とするコンピュータシステムを最終的に

品質保証するために，日立製作所が１９７８年以来実施してきたＳＳＴの概要を述べる．

11.2.1　ＳＳＴの目的

SST は顧客の品質保証に対する要求に応えるために開始された．SST の基本的な目的は，日立製作所の工場内で実際の顧客環境で発生する負荷やストレスをシミュレートし，システムの出荷前に予想される不具合をすべて摘出し修正することである．

SST には，主な３つの目的がある．

（１）顧客と同等なシステム構成下で，顧客業務の機能や性能目標を満足することを保証すること
（２）負荷の変動や機器の障害に耐えられるフォールトトレラントシステムであることを保証すること
（３）稼動後のシステムの変更や拡張により，運用や性能面で不具合が発生しないことを検証すること

11.2.2　システム開発におけるＳＳＴの位置づけ

システム開発プロセスとＳＳＴの位置づけを，図１１．１に示す．

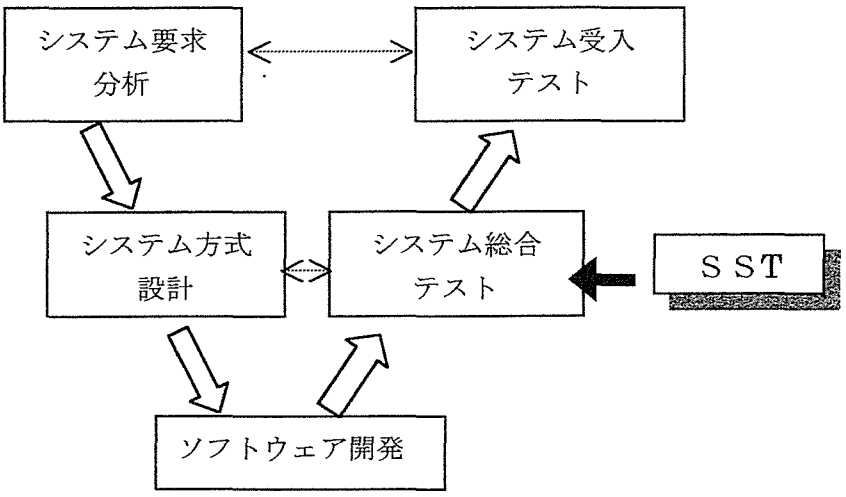


図１１．１　システム開発におけるＳＳＴの位置づけ

ＳＳＴは，情報システム開発プロセスのシステムテストフェーズの一部である．ソフトウェア開発が終了し，独立Ｖ＆Ｖ（Verification　& Validation）合格後，ＳＳＴは，顧客のサイトでの実際のシステム運用に先だって，ハードウェア製品や，

他社のソフトウェア製品を統合したシステムとして日立製作所内のＳＳＴセンタで実施されるシステムテストの一形態である．これは高信頼性でディペンダブルなシステムを実現するための重要なステップである．ＳＳＴ実施後，通常は顧客サイトでシステムの運用テストを行い，その後運用を開始する．

11.2.3　ＳＳＴの定義

ＳＳＴとは，ハードウェア，ネットワーク，基本ソフトウェア，業務プログラムからなる顧客システムを，供給者（日立製作所）の施設内に建設し，顧客の運用状態にできるだけ近い形態で，その全体，または一部の運用を擬似的に試みることにより，顧客への納入に先だって，システム総合機能の正常性を日立製作所として確認することである．

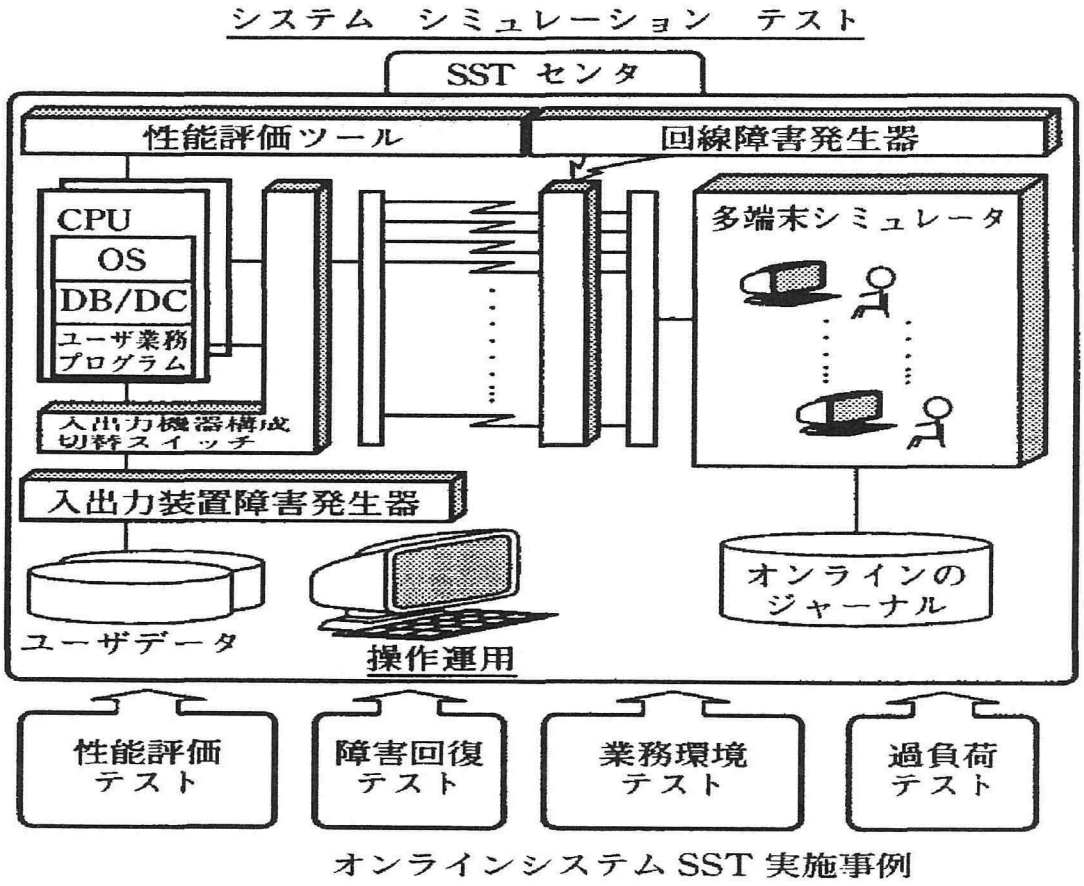


図１１．２　ＳＳＴの概念

11.2.4　ＳＳＴのコンセプト

図１１．２は，オンラインシステムに対する単純化したＳＳＴの概念を示している．図中の上部のシステム構成図は，顧客システム環境をシミュレートしたテスト対象システムである．これは，ハードウェア環境およびソフトウェア環境を，実際の顧客システム環境に極力近づけるようにシミュレートしたものにする必要がある．例えば，多数の端末を用意して実際のオペレーションをすることは困難であるため，独自に開発した多端末シミュレータを使用する．擬似顧客システム環境に対して，下辺の矢印で示したのはテスト用の入力である．正常業務用入力データと，業務運用オペレーション，システムへの過負荷を与える各種の入出力データ／トランザクション等，システム障害の原因となるソフトウェア／ハードウェア障害／オペレーションミス等のストレスを，あらかじめ用意したテストケースに基づいて加える．その下にはSSTの主要なテスト種別を記してある．

SSTは，個々のシステム構成要素にではなく，システム全体の運用に着目する．SSTのテスト種別は，運用，障害回復，システム性能の３つである．SSTを実施するＳＥは，当該顧客システムに適したテスト項目を各テスト種別から選択する．

11.3　ＳＳＴの実現方法

本節では，１１．２の内容を具体化する手段としてのSSTの設備とツールを紹介する．さらに，テストケースの例についても述べる．

11.3.1　SSTの設備

日立製作所は，シミュレーションテストのための次のような専用の設備をSSTセンタに用意している．

(1)多岐にわたる顧客の業務システム構成を構築できる標準的コンピュータハードウェア／ネットワーク設備（ＣＰＵ，サーバ，クライアント，ディスク他各種入出力装置，端末装置，ルータ，通信制御装置，ＬＡＮ，ＷＡＮ等のネットワーク環境等）
(2)あらゆる種類の顧客システムに合わせて，SSTセンタ内のハードウェア構成を迅速かつ効率良く，変更するための特注切り換えスイッチ類
(3)シミュレーションテストの状態を常に監視できるための各種計測装置類

11.3.2　SSTのツール

ＳＳＴには下記の３種類のタイプのツールを用意して，SSTのテスト精度の向上と効率向上を実現している．

(1)テスト環境の構築のためのツール
- 実際の構成環境の代わりに使用される各種シミュレータ
  (端末シミュレータ，ネットワークシミュレータ，大容量ディスクシミュレータ等)

(2)システムテストの実施を支援するためのツール
- 実際のあるいは将来起こり得る負荷に近い環境条件の実現や性能テストを実施するツール
  (多端末シミュレータ，性能評価ツール等)
- 各種デバイスやネットワークのエラー条件をシミュレーションするエラーシミュレータ
- エラー原因の解析支援ツール

(3)テスト結果の検証
- 処理結果の正当性を効率よくチェックするツール
  (テスト結果比較判定ツール等)
- 性能評価のための解析支援ツール

11.3.3 テストケースの例

オンラインシステムのＳＳＴで要求されるテストケースの例を以下に示す.

- システム障害後のリスタート
- あらゆるハードウェア障害後のシステムおよびソフトウェアの回復
- 大量のオンライントランザクションの待ち合わせと処理
- 共有資源の並行処理
- データベースの保守
- オンライン処理中のボリュームコピーやマージ処理
- オンラインジョブとバッチ（一括処理）ジョブの同時処理

テスト結果を解析するとき，ＳＥはシステム設計時に計算した理論値と測定値を比較する．その結果，不一致がある場合には，その理由を調査し，使用者が許容できるかどうか判定する.

11.4 ＳＳＴの実施方法

11.4.1 ＳＳＴの実施体制

日立製作所は，このシステムテストを推進するために，横断的組織をもっている．システムＱＡ部門がＳＳＴセンタの運営とシステムテストを推進するＳＳＴ

グループをもち，当該受注システムを担当するＳＥ部門および各ソフトウェア製品，ハードウェア製品開発部門とプロジェクトを組んでＳＳＴを推進している．

さらに，必要に応じて顧客のデータを借用したり，顧客自身に立ち会ってもらうこともある．これにより，顧客は，納入以前にシステムの動作状況を確認でき，信頼感を増すことができる．ＳＳＴグループは，ＳＳＴセンタのテスト機器の導入計画や，各種テスト用ツールの開発導入も行う．

図１１．３に，ＳＳＴの実施時期と体制を示す．図１１．３の左側の各工場の関係者とＳＥが主体になり，必要に応じて当該顧客もファイルやプログラムの貸し出しや，運用オペレーションに参加する．

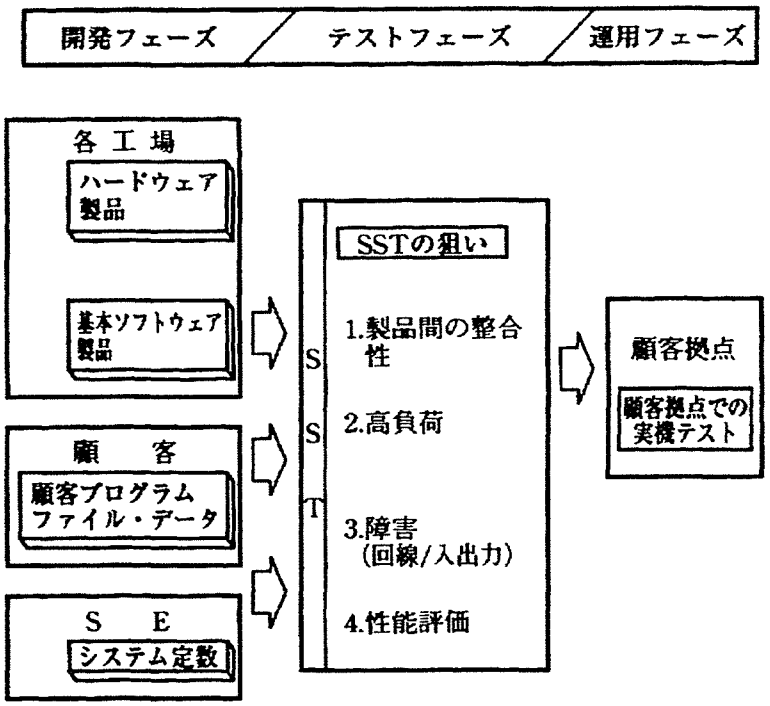


図１１．３　ＳＳＴの実施時期と体制

11.4.2　ＳＳＴ対象システムの選定と実施結果のフォロー

営業，顧客担当ＳＥ，製品担当工場・事業部等の関連部署から構成されるＳＳＴ委員会が，どの顧客受注システムをＳＳＴの対象システムにするか，基準に従って決定し，テスト方針の審議を行う．

ＳＳＴ実施基準は，以下の２つの要因を組み合わせて構成される．

・当該システムのトラブルによる社会的影響度
・当該システム開発の技術的難易度（新製品・システム，新技術，新サービス，未使用他社ベンダー製品，各製品の初組合せ等）

ＳＳＴ実施後には，ＳＳＴ実施結果の評価を行う．

11.5　ＳＳＴの変遷

図１１．４は，ＳＳＴの主たる対象となるシステムの変遷を示したものである．

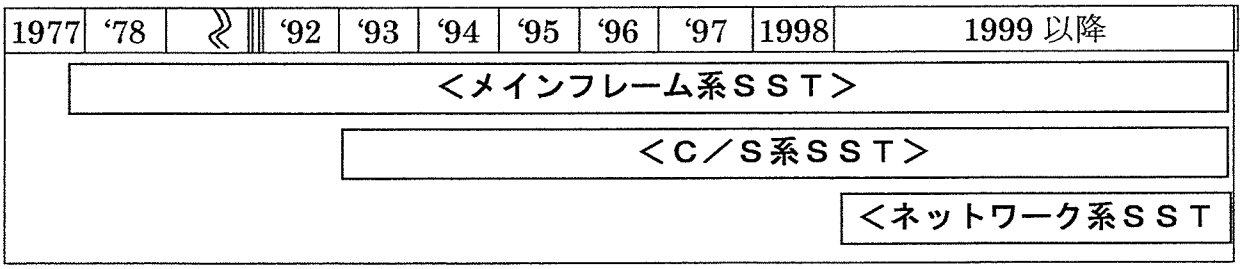


図１１．４　ＳＳＴ対象システムの変遷

（1）メインフレームコンピュータ全盛期（1977-1992）

この期間は日本経済の高度成長期にあたり，専売権をもつメインフレームコンピュータ全盛期であった．したがって，ＳＳＴの対象システムは大半が日立製品で構成された，かつ公共性の高い大規模なオンラインリアルタイムシステムが主体であった．ＳＳＴの焦点は，大規模システム構成の構築と高信頼性の保証であった．

（2）クライアントサーバーシステム（1993-）

コンピュータのダウンサイジング化とオープン化により，クライアントサーバーシステムが急速に進行した時代である．バブル経済が崩壊し，低価格で，ハードウェア，ソフトウェア共他社製品による，ブラックボックス製品群によるシステム構築が主体であった．ＳＳＴの焦点は，多様な他社製品の手配と製品間の整合性の確認である．

（3）ネットワークシステム（1998-）

ＬＡＮやＷＡＮさらにインターネットの普及により，様々な形態のネットワークシステムの構築が主体であった．ＳＳＴの焦点は，多岐にわたるネットワー

クの信頼性と，"Best Effort"製品を含むシステム性能の検証である．

11.6 大規模ネットワークシステムへの対応

１１．５で述べたように最近は大規模ネットワークシステムに対するＳＳＴが増加しているが，このようなシステムに対して，１１．４までに説明したＳＳＴの考え方がどのように実現されているかについて，以下に述べる．

11.6.1 ネットワークシステムのＳＳＴが増加している背景

最近，ネットワークシステムの大規模化と複雑化が進み，さらに受注が増加したため，稼動後にこれらのシステムのネットワーク障害が急増した．

これらの障害発生原因のひとつは，ネットワーク設計が不十分なためである．これは，ネットワーク設計体制が受注の増加に追いつかないことと．最新のネットワーク技術力不足およびノウハウの不足がある．

もうひとつの原因は，ネットワークシステムの検証不十分である．これに対して，ＳＳＴで実施した改善対策を以下に述べる．

11.6.2 ＳＳＴにおけるネットワークシステムへの強化策

(1) ＳＳＴセンタ設備の強化

(a) 設備面

業務アプリケーションとネットワークシステム（ホスト系—サーバ系—ネットワーク系—端末系）の連動確認に必要な不足機器の増強を行った．

(b) ツール面

各種テスト支援ツール（負荷発生シミュレータ，障害発生シミュレータ，検証用ツール，網シミュレータ）の充実を図った．

(2) システムテスト・ノウハウ集やガイドラインの整備

「システムテストの着眼点」の体系は以下に示す通りである．これに対して，上記原因分析から得られたノウハウを追加した．

●システムテストの着眼点の体系

○システムテスト共通編

○テスト種別編

・運用（通常・障害）テスト

・構成（環境）テスト

・性能テスト

・負荷テスト［処理の交錯・競合タイミング］

○システム構成編
・C／Sシステム
・ネットワーク
・インターネット

11.7 大規模銀行システムのＳＳＴ事例

11.7.1 システムの狙い

このシステムは，以下の２点を目的として構築された．

・マルチサーバ構成の分散C／Sシステムによるトランザクション処理能力の向上
・ホストシステムからオープンシステム携帯への移行によるシステム変更時の柔軟な対応

11.7.2 ＳＳＴの目的

各支店システムとネットワークに対する目的は次の通りである．

(1) 各支店システム
　各サーバの性能および障害時運用確認と，システム（ホストとサーバ連動）の性能及び障害時運用確認
(2) ネットワークシステム
　業務を稼動させての全店稼働時の性能，障害時のリルート（re-root）確認，正常時の経路確認及びネットワークの負荷耐性・限界性能の確認

11.7.3 ＳＳＴの方式

ＳＳＴでは，本番システム構成を縮小した構成を構築し，多端末シミュレータを用いて高負荷を発生させた状況下で，性能をはじめＳＳＴの目的で掲げた検証を実施した．図１１．５に大規模銀行システムのＳＳＴ事例を示す．

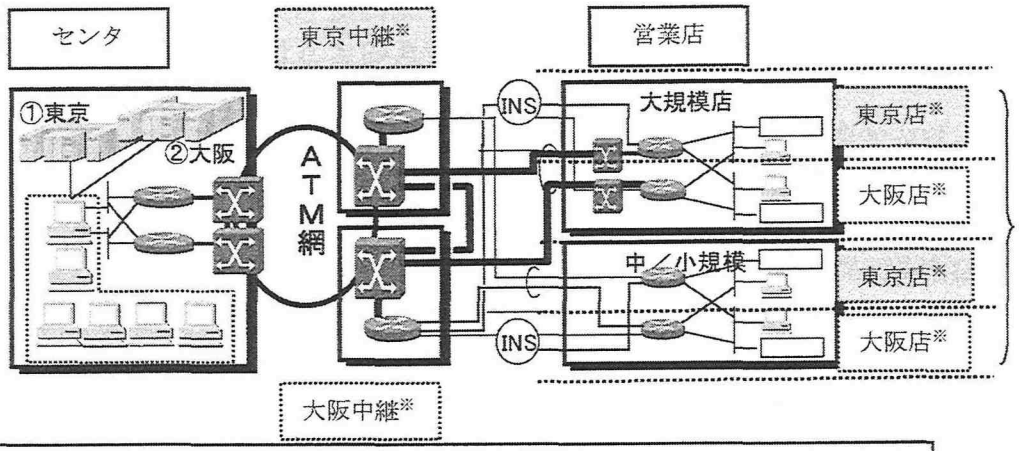


図１１．５　大規模銀行システムのＳＳＴ事例（ATM：５０００台／３００店，新端末：４０００台／３００店相当からの負荷発生）

ネットワーク系のＳＳＴ摘出不良の内訳は，図１１．６の通りである．

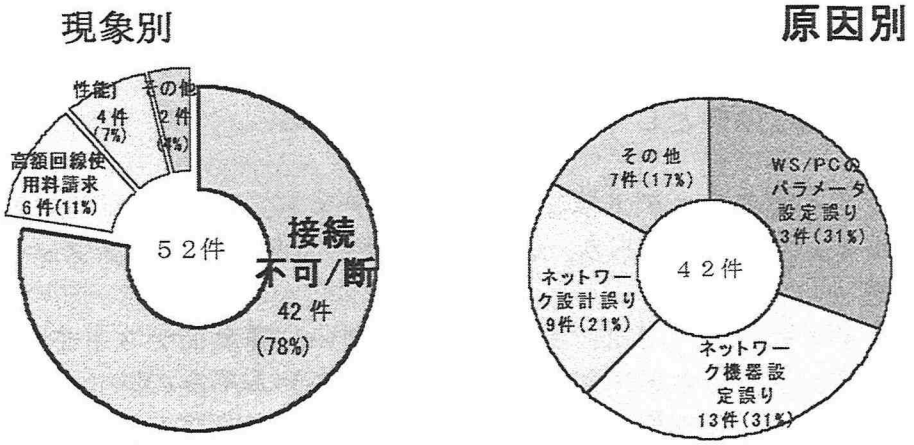


図１１．６　ＳＳＴ摘出不良の内訳

11.8　成果

これまで述べてきたように，ＳＳＴは，情報システムの急速な変貌に対して敏速に対応してきた．その結果，以下のような成果をあげており，顧客の高い信頼を得ている．

（1）最大の課題であったネットワーク系システムに対しても，11.6および11.7で述べたような施策により，質の高いテストができるようになった．

（2）ネットワーク系のＳＳＴ不良摘出率は，図11.７に示す通り，１９９０年は７７％であったが，その後年々向上し，１９９６年以降は９９％を維持している．ここで，ＳＳＴ不良摘出率＝ＳＳＴ不良摘出件数／（ＳＳＴ不良摘出件数＋ＳＳＴ見逃し不良件数）である．

なお，１９９０年代前半に摘出率が向上したのは，以下の理由である．それ以前は内容の分かっている自社製品の組合せのシステムであり，ＳＳＴでの不良摘出は少なかったのに対し，１９９０年代前半になり，他社製品の組合せが増大し，ＳＳＴでの不良摘出が増加した．そのため，ＳＳＴ不良摘出率が向上した．

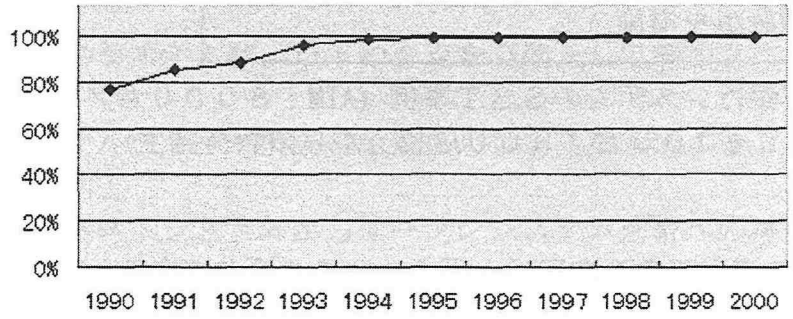


図１１．７　ＳＳＴ不良摘出率の推移

ＳＳＴの定性的な効果としては，以下があげられる．

（1）顧客サイトでのシステム構築，受け入れテスト，顧客稼動段階での初期トラブル（システム構成不良，マルチベンダー製品間のインタフェース不良）の激減
（2）顧客サイトのテストでは，実現が容易でない，高負荷テストや各種のハードウェア・ソフトウェア障害に起因するシステム不良の防止
（3）顧客納入前に，日立製作所内のＳＳＴ設備で顧客業務が稼動することを確認できることによる顧客信頼感の増大

11.9　まとめ

情報システムは，１９９０年代以降オープン化やインターネットを含むネットワーク化により，大きな変貌を遂げてきた．一方で，情報システム構築の環境は，短納期化，低価格化が進み，一段と厳しくなっている．さらに信頼性を要求されるシステムの構築にあたっては，以下のような厳しい条件にさらされている．

（1）「そこそこ品質」(good enough quality) のＰＣソフトウェアや最良の努力はするが保証はしないという「ベストエフォート型」(best effort)通信サービスの出現
（2）オープン化により，ブラックボックス仕様製品の組合せによるシステム構築

このような条件の変化は，ＳＳＴというシステム品質の保証活動を難しくしてきたが，我々はそうした厳しい条件を一つ一つ克服し，四半世紀にわたる良き伝統を継承している．見方を変えれば，上記の環境変化は，ますますＳＳＴアプローチというシステム指向が重要なものになりつつあるあることを示唆している．

１１．８で述べた通り，ＳＳＴを実施した情報システムの稼動後のシステム信頼性は高い水準を維持している．正に「継続は力なり」である．

第１２章　上級ソフトウェア技術者実践教育ＳＥＰによるスキルの継承と向上

12.1 はじめに

1980年代後半頃から,新しいソフトウェア生産技術が開発現場でも必要とされるようになってきた．このような時代背景の下に，コンピュータ・メーカを中心として，上級ソフトウェア技術者育成が強く求められるようになった．一般に，上級ソフトウェア技術者は集合教育などでの育成は難しく，OJT（On the Job Training）で育てざるを得ない，との意見が業界の常識であった．しかし，技術革新の激しいこの分野では，徒弟制的なOJTに依存した育成だけでは，時代に即応したプロフェッショナル技術者集団は作れないと判断し，逸速く上級ソフトウェア技術者の育成を目的として，日立製作所の技術研修所でSEP（Software Engineering Project）コースを開発し，実践することにした．

ソフトウェア・エンジニアリング分野でのロール・プレイング式教育は，欧米でもいくつか試みられている．とくにSEPと類似の教育では，カナダのトロント大学での例が最も旧く，最近ではイギリスのGTP社，ロックバラ，ダービー両大学での実施報告がある[40]．トロント大学の特徴は，プログラミング教育であり，チーム編成で作品を競い合うゲーム方式を取入れた点である．GPT社，ロックバラ，ダービー両大学の例は，実施時間が70～80時間程度であることと，設計技術に焦点を当て，管理技術は対象にしていない．日本では，三菱電機での実施例を大学教育に適用した例が報告されている[41]．この実施例の特徴は,「ソフトウェア設計方法論」に主眼を置き．企業で大規模なソフトウェアを制作するときの方法論を工学部の学生に15回の講座で教えた実践報告である．しかしながら，SEPが狙っている，開発技術と管理技術の両面を一体化した長期間の教育については報告事例がない．

さらに，近年，技術者教育認定の必要性から，アメリカの大学でもソフトウェア・エンジニアリング教育の体系的なカリキュラムが産官学共同で整備されつつある[42]．日本でも，1999年に発足したJABEE(日本技術者教育認定機構)が推進母体となって，CS (コンピュータ・サイエンス)，SE (ソフトウェア・エンジニアリング)，IS (インフォメーション・システム)の３分野で，大学の学部教育カリキュラムの整備が進められている．特に，ソフトウェア・エンジニアリングの分野では，プロジェクト・マネジメント（管理技術）が注目されている．さらに，大学の情報系学部において，技術者教育認定制度とも関連して上級ソフトウェア技術者育成カリキュラムが盛んに議論されている折から[43]，ここに力点を置いたSEPが注目されている．

本章では，従来集合教育では難しいとされた，上級ソフトウェア技術者教育の考え方を述べる．1990年のSEPコース開設以来，10年間の実績を踏まえ，その概要と評価結果について述べる．特に評価については，受講生の知識獲得と実務能力について，OJTとの比較結果を述べる．また，SEP受講直後と受講後２年および３年を経過した時点での有用性調査を報告する．

12.2 SEPコースの開発方針

SEPコース開発の目的は，与えられた業務の対象領域において，市場や顧客の要求を満たすソフトウェアを，必要最小限のリソースを使って製品開発するための，実践的な技術を身に付けさせることにある．細かな修得すべき事柄は，技術の進歩とともに変化していくので，様々な要請を満たした基本的な技術や，システム設計の基本プロセスを確実に修得させることにした．ただし，カリキュラムや演習内容は毎年見直し，洗練化を図っている．図１２．１に教育コースのカリキュラムを設計するためのソフトウェア・エンジニアリング教育の概念モデルを示す[39].

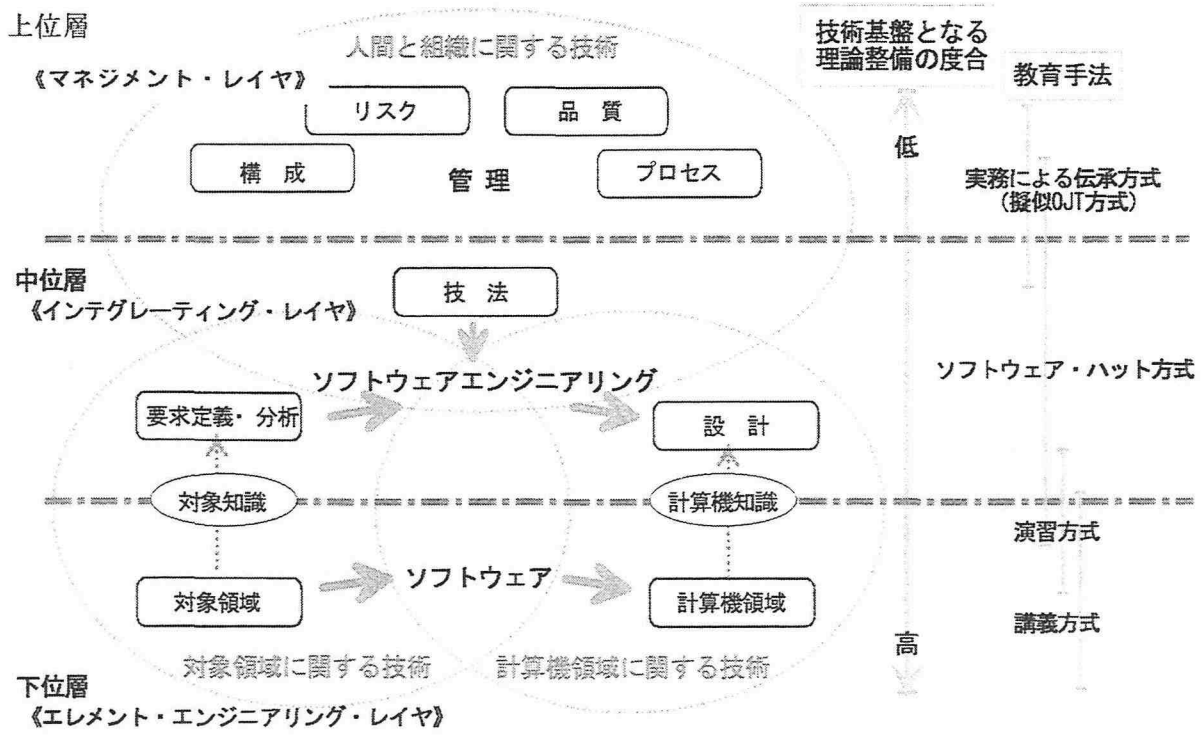


図１２．１　ソフトウェア・エンジニアリング教育の概念モデル

概念モデルの下位層は，エレメント・エンジニアリング・レイヤで，ソフトウェア技術そのものの枠組みを表している．これは，ソフトウェア本来の役割が「対象領域の問題をコンピュータを使って解く」ことを端的に示している．この領域は，コンピュータの発展そのものを支えてきた情報科学・情報工学の中核であり，技術基盤となる理論整備の度合いも高く，講義・演習方式もほぼ確立されている．

中位層は，インテグレーティング・レイヤで，ソフトウェア・エンジニアリングの技術の枠組みを表している．ソフトウェア・エンジニアリングは，ソフトウェアを対象としたメタ技術である．下位層のソフトウェア技術の体系と中位層のソフトウェア・エンジニアリング技術の体系とは強い関係にある．ソフトウェア・エンジニアリングの技術は，対象領域の知識を定式化・モデル化し，それに基づいてソフトウェア・システムの要求を抽出する要求定義・分析技術，コンピュータへの実装上の制約条件のなかで要求を満たす実現方法を決定する設計技術，さらに，これらの分析と意思決定を人間と組織が的確に行えるようにするための技法などから構成される．この領域での技術基盤となる理論は，いまだに十分揃っているとはいえず．理論体系も明確になっていない．ソフトウェア生産技法として実務指向の手順や方法論は提唱されている．近年，米国国防総省の支援の下に，大学や学術領域でのカリキュラム体系の整備に拍車がかかっているが，この領域の教育手法に関する効果的な実施例は報告されていない．中位層の技術伝承を行う教育手法は，一種のロール・プレイング方式として後述のソフトウェア・ハット方式を導入し，理論整備が不十分な世界であるものの，体系的なソフトウェア生産技術の修得を行う必要がある．

上位層は，マネジメント・レイヤで，ソフトウェア・エンジニアリングの技術で使われる，諸々の資源を管理する技術の枠組みである．企業でソフトウェア（プログラム）を製品として製作する場合，発生する原価を細かく把握し，工程遅延を生じさせず，品質（信頼性）を高め，予期せぬ出来事を可能な限り未然に防ぐリスク管理を徹底する必要がある．これらを，人間と組織に関する「管理技術」と呼ぶ．この領域は，技術基盤となる理論整備の度合いが低く，実務による伝承方式，いわゆるOJT方式が根強く残っている．しかしながら，上位層の技術を単純に切り離してOJT方式だけで教育を行っても，それぞれの管理対象は中位層や下位層に及ぶため意味がない．ソフトウェア・ハット方式の中で，あるいは，これと強く連携した演習方式を取り込み，効果的な管理技術の修得を目指す必要がある．表１２．１では，これを「疑似OJT方式」と表記してある．

図１２．１に示した教育の概念モデルを作業仮説とし，上級ソフトウェア技術者のために必要な内容と修得のレベルを設定して，SEPコースのカリキュラムを編成している．

概念モデルの下位層の「ソフトウェア技術」に関しては，ソフトウェアそのものの対象領域や計算機領域の事項については，比較的に変化が激しいため，時

宜を得たものに設定する必要がある．これは，演習の課題設定を変えることで対処している．

中位層の「ソフトウェア・エンジニアリング」に関する技術は，「開発技術」に関する講義と演習を中心に行う．プロフェッショナルには，技術的な知識だけでなく，それらを組み合わせて，正しく稼動するシステムを設計し，確認することが要求される．このため，講義と演習を一組に設定し，使いこなせるまで技術力を向上させなければ，実用的・効果的な訓練にはならない．

上位層の「管理技術」については，「開発技術」以上に，実際にやらせてみて，管理者としての素養を身につけさせなければ意味がない．このため，講義だけではなく，「ソフトウェア・ハット」と呼ぶ実践的な「プロジェクト演習」を採り入れている．そのさい，国内における開発プロジェクトの改善を効率的に展開するために教授項目に優先順位をつけ，さらに実務展開のための品質評価手法などを導入した．また，成果物の交換など開発プロセスには新規の手法を考案した．講義内容は，国際標準を考慮して，IEEE Standard 1058 (Template for a Software Project Management Plan) [44]に準拠させてある．

講義方式による技術の伝承は，受講生に学習の契機を与えるのが目的である．したがって，受講生の学習に関連した活動としては，講義受講だけでは意味がなく，演習への参加，必要な教材の予習と復習，プロジェクト演習の準備，メンバー間や演習講師（チュータ）との連絡，グループ討議，質疑応答，レポート作成，発表資料作成，プログラミングなど，多岐に亘るようにしてある．このため，SEPでは，講義時間以外について，時間や場所に束縛されない学習環境を整備した．例えば，ネットワークを介しての指導・連絡などはこの具体的事例である．

12.3 SEPコースの特徴

12.3.1 SEPコースの特徴と目標

SEPコースの第１の特徴は，要求仕様を定義する工程（上流工程）から，設計・製造（プログラミング）工程を経て保守工程（下流工程）までを「連続した流れ」のなかで学習させることにある．これは，企業内でソフトウェア生産を進めるときの展開を，そのまま体得させる必要があることを意味する．技術の体得には時間がかかるため，実業務と並行して，教育期間は１年とし，隔週２～５日の集合宿泊形式を採用した．教育のために拘束する期間を，1999年は全体で51日間にした．このほかに，各人が予習復習など前述の作業に相当な時間を振り向けられるようにした．

第２の特徴は，先進的な設計技法などの「開発技術」と，前述の「管理技術」

の体得に均等の時間を配分していることである．この理由は，企業でソフトウェアを生産する場合，多数の人間が参画することと，外的環境の変化など不確定要因が発生するため，原価・工程・品質などのほかリスクに対する管理技術も非常に重要なためである．

第３は，全時間の半分強を「ソフトウェア・ハット（Software Hut）*」と呼ぶプロジェクト演習に配分していることである．これは，講義で学んだ開発技術や管理技術を駆使して，実際にプログラムを作成する訓練である．企業のソフトウェア開発現場では，１人のリーダが数人の担当者を指導しながら仕事を進めるのが標準的な生産単位となっている．そのため，企業内の組織的な活動の模擬実験となるようにするため．４人１組のチームで開発に当たらせる仕組みにしてある．「ソフトウェア・ハット」については，１２．３．２で述べる．

第４は，演習指導には「チュータ」と呼ぶ演習講師を配備したことである．チュータは，チームをソフトウェア生産企業に見立てた場合の，経営コンサルタントの役割を果たす．また，併せて，徒弟制の親方（匠）の役割も果たすことになる．チュータは，コース開設２年目以降は，SEPコースの修了者を当てることにした．上級ソフトウェア技術者を育成する上で，非常に重要な役割を演じると同時に，SEPコース修了者として一層のレベルアップに役立てることも期待した．チュータ制について，詳しくは１２．３．３で述べる．

第５は，開発技法やプロジェクト管理の講義・演習とソフトウェア・ハットの指導に，米国の２人の教授，フェアリ(R. E. Fairley)教授とゴマ(H.Gomma)教授を招聘していることである．この理由は２つある，第１は，ソフトウェア・エンジニアリングの教育分野では，理論と実践の両面を担当できる指導者は，日本では適任者が見当たらなかったためである．第２は，技術者の国際化にはネイティブによる英語教育が必須となるためである．両教授は，SEPコース開設当初からの良き理解者である[45]．これらの講義・演習とソフトウェア・ハットは，すべて英語で実施するため，使用する教科書の予習を全員に義務づけ，受講中も積極的に質問させ，チーム内での復習の徹底など，英語力のバラツキが障壁にならないような工夫も施した．

以上の特徴を有効に機能させるため，受講生は，事業所長または関連会社社長の推薦を受けた者から選定する．推薦の基準はつぎのとおりである．

（１）受講者自身が積極的な学習意欲をもち，本教育の厳しい負荷に耐えうること．
（２）原則として，開発・設計および品質保証部門を担当する者．
（３）大学の学部卒業以上，あるいはそれと同等の実力を有する者．
（４）５年程度のソフトウェア開発の実務経験を有する者．
（５）数百行のプログラムを数本以上書いた経験があること．

---

* 「ソフトウェア・ハット」は，ハットの意味が小屋・工房なので，ソフトウェア・ハウスよりも小規模な開発組織を意味する．

（6）C/C++/Javaの言語の基礎知識を有する者.
（7）フェアリ教授とゴマ教授の講義・演習は英語で行うので，それに必要な英語力を有すること.

これらの基準で選抜された受講生を，４人１組にしたチーム編成にした．事前アンケートと面談の結果で，各チームのバランスがとれるようにチーム編成とリーダを暫定的に決めるが，プロジェクトの進行とともに，メンバーの協議によるリーダ再選出と，チーム間でのメンバー交換を許している．これは，各チームを１つのソフトウェア会社に見立て，「計画したソフトウェアを確実に開発する」ため，日常業務で行われている人的トレードを，そのまま同じ環境で実行できるようにすることを狙ったものである．業務の都合や個人的な事情で集合教育に参加できない人も出てくるが，リーダはこれを「リスク管理」として織り込んでおかなければならない．「与えられたリソースを有効に使い所期の目的を達成する」ことは，通常業務でもリーダが責任をもって遂行しなければならない重要な要件である．このため，経営コンサルタントである「チューター」を巧く使い，他のチーム（競合他社）の状況把握や情報交換を頻繁に行うことの重要性を学ぶことになる．

SEPコースの教育効果は，つぎの４点を目標にした．

（1）事前テストでの平均値未満の成績下位群を，徹底的に指導して，飛躍的に伸ばし，全体のレベル向上を図る．
（2）才能のある者は最大限その才能を伸ばす．
（3）受講生だけではなく，チューターにも，より深く学ばせることにより，さらに全体のレベル向上を図る．
（4）米人教授による講義・演習を体験させ，英語力のバラツキによる影響力を少なくする環境を整える．

12.3.2 ソフトウェア・ハット

企業ではビジネス上の制約から，的確な時期に製品を提供する必要がある．このため，必要な機能を小刻みに実装するインクリメンタル開発プロセスを採用する．開発チームの体制も，チーフ・プログラマ制ではなく，SWAT (Skilled With Advanced Tools)チームという少数精鋭型やTIT (Thread Integration Team)と呼ばれる主副デザイナー方式など多様化させている[39].

マネジメントに関わる知見の修得は，理論や原理を学んだだけでは不充分である．しかし，こういった事柄をすべてOJTで行うには限界があり，なんらかの疑似体験型の学習方式を採り入れなくては，実践的な効果は期待できない．この学習方式が，「ソフトウェア・ハット」方式である．講義・演習とソフトウェア・ハットとの関係を，表１２．１に示す．

ソフトウェアの原理的な事項や開発技法に関しては，ある程度，座学で修得

できるが，管理については，教育技術上，さらなる工夫が必要になる．ソフトウェア・エンジニアリングの管理には，構成管理，プロセス管理，品質管理，リスク管理など，製品を開発するのに欠かせない事項が多い．これらの項目を含むプロジェクト計画立案を具体的に行わせる演習方式を採り入れている．演習課題は現実にある情報処理システムや，制御システムの開発を想定した疑似体験型の学習方式であり，表１２．１では，「プロジェクト計画演習」と表記してある．1999年度のコースでは，プロジェクト計画立案に関する講義と演習に対して，１学期（3ヶ月）を割り当てている．

プロジェクト計画は，組織編成，成果物の仕様，品質，工程，リスク，使用ツールなど，あらゆる観点を盛り込み，それに従って実際にソフトウェアを開発し，計画の妥当性を含めて評価を行う．

「プロジェクト計画演習」がプロジェクト計画の立案に主体を置いているのに対して，計画に沿った開発とプロジェクトの監視・評価を行う方式の演習が，表１２．１の「設計演習」，「コーディング・テスト演習」，および「機能追加演習」である．個人で創る趣味のものではなく，製品開発を行うときに必要な事柄は，計画を立て，これを組織的に遂行することである．ソフトウェア・ハット方式は，小規模のソフトウェア・ハウスやSWATチームを想定したロール・プレイング・ゲームを採り入れた指導方法である．

表１２．１　講義・演習とソフトウェア・ハット

学　　期	講 義・演 習	ソフトウェア・ハット
第１学期 11.5 日 （３ヶ月）	・**Project Management**（**3.5** 日）英語 (Overview, Planning and Estimation) ・品質管理（１）（0.5 日）	・ガイダンス（1 日） ・プロジェクト計画演習（5.5 日） ・第一学期発表会（1 日）
第２学期 19 日 （３ヶ月）	・**Development Method**（5 日）英語 ・オブジェクト指向開発技法（3 日） ・品質管理（２）（0.5 日） ・設計論（3 日）	・設計演習（6.5 日） ・第二学期発表会と製品交換（1 日）
第３学期 13 日 （３ヶ月）	・**Project Management**（**4** 日）英語 (Monitoring, Controlling and Risk Management) ・クリーンルーム手法（0.5 日） ・特別講義（１）（0.5 日）	・コーディング・テスト演習（7 日） ・第三学期発表会と製品交換（1 日）
第４学期 7.5 日 （３ヶ月）	・品質管理（３）（0.5 日） ・特別講義（２）（0.5 日）	・機能追加演習（6 日） ・修了発表会（0.5 日）

受講者を４名からなるチームに分け，それぞれのチームを１つの小さな組織体（企業）に見立てる．それぞれのチームは，製品の計画から開発までを何回も行う．このとき，他チームの成果物を取り込んだり，他チーム（他社）へ提供したりしながら開発を進めて行く．表１２．１では，「製品交換」と表記してある．この製品交換にともなって，製品関連情報を記述した各種のドキュメンテーションも提供され，「わかりやすいドキュメンテーション」の重要なことを認識させることになる．

１2.3.3　チュータの選定と役割

SEPの目的は，ソフトウェア製品開発をするための実践的技術を修得させることである．このため，教授の講義内容や演習方式を熟知した助手の存在が不可欠である．この助手つまり演習講師を，SEPでは「チュータ」と呼ぶ．チュータは，毎年，SEPコース終了時点で，「技術力・理解力・指導力・忍耐力・協調性」の５項目を総合評価して，３名を目安に選抜する．チュータは，１チーム当たり１人の割合になるようにするため，全体で最少６名を必要とする．所属元との約束で，チュータの在任期間は２年間に決めてある．指導内容の連続性を考慮して，任期での交代は，半分の３名とするために，１年ずつずらせてある．したがって，１年間のコースが修了すると，任期満了の３名のチュータが所属元に戻り，補充の３名を，新修了者から選抜する仕組みにしてある．1990年から1999年までに37名が任命された．チュータにとって，在任期間の２年間は「教えることを通して，より多くを学ぶ」機会でもある．SEPコースにおけるチュータの役割は大きく，SEPコースの質を高めるには，チュータの質を高めることが必須となる．チュータの役割は，技術的な指導と，プロジェクト・マネジメントとに大別できる．1997年までは，１チームに１人の専属チュータを配置したが，1998年からは，専門分化し，テクニカル・グループとマネジメント・グループに分け，集団で全チームの指導に当たらせるように改善した．これは，「技術力がアンバランス」，「チュータの意思疎通が不充分」，「チュータを選べるようにしてほしい」という受講生からの改善提案への対応策でもあった．この機能対応策は，チュータ選抜にも選択肢の幅を広げる効果があっただけではなく，運営にも柔軟性が増す効果もあった．

12.4　評価

SEP コースの特徴をまとめると，次のとおりである．

① 上級のソフトウェア技術者を養成する.
② 教育の対象は，図１２．１の下位層《エレメント・エンジニアリング・レイヤ》から，上位層の《マネジメント・レイヤ》まで幅広い.
③ 学習方法は，講義と演習のほか，ソフトウェア・ハット方式や，実務による伝承を取入れた《疑似OJT方式》を組み合わせている.
④ 知識だけでなく，実務遂行能力の向上や，職場の変革者としてのリーダを育成する.
⑤ グローバル化に対応できるよう，英語での授業を相当量取入れる.
⑥ 教育期間が１年間と長く，集合教育と，予習・復習および職場での実務とを一体化して学習ができるように配慮する.

以上のように，高度で広範囲かつ多目的な教育を目指したものであるため，単一の物差しで評価するのは適切でない.

そこで，できるだけ総合的に評価できるようにし，まず知識獲得や実務遂行能力を客観的・定量的に測定する評価尺度を設定し，SEP受講者群とOJT群との比較を行った.

次に，受講者自身がSEPコースをどのように評価したかをアンケートで調べた．さらに，受講後２年経過者と３年経過者についてもアンケートを行い，実務を経験した上でのSEPに対する評価推移を調べた．そして，英語での講義が学習成績に影響を与えたかどうかを評価した.

12.4.1 知識獲得と実務遂行能力の評価

事前テストと事後テストは，対象者全員がソフトウェアの知識を有しているために，「同一問題」を採用した.「同一でも意味・意義がある」問題として表１３．２に示す25個のキーワードを設定し，この用語の理解度が，事前と事後とでどう変化するかを調査した．この25項目は，単なる知識ではなく，知恵と経験を深めることによって身に付くソフトウェア・エンジニアリング分野での重要概念とし，論述式とした.そして，全チューターから50個を厳選してもらい，フェアリ，ゴマ両教授を交えたコーディネータ会議で，普遍性などの観点から，さらに25個に絞り込んだ．各項目の理解度を０～４の５段階に分け，各問を下記の基準で判定した.

０点：理解していない
１点：知識としてはほぼ理解している
２点：知識としては完璧に理解している
３点：他人に説明でき，使いこなせる
４点：あらゆる場面で応用できる

５段階評価は，ACM（Association for Computing Machinery）のno knowledge, recognition, literacy, usage, applicationに準拠して，客観的・定量的に判定できるようにした[46].

なお，事後テストは修了試験ではなく，あくまでも「上級ソフトウェア技術者に求められる重要概念の理解度や，それに基づく実戦力・変革力」が目的のため，判定は，チュータ全員の合議制で厳しく査定した．

表１２．２　25個のキーワード

No.	キーワード	No.	キーワード	No.	キーワード
1	リスク管理	10	要求分析・定義	19	CORBA
2	スパイラルモデル	11	クラス／オブジェクト	20	コンポーネントウェア
3	統計的品質管理	12	デザインパターン	21	要求工学
4	OMT法	13	段階的詳細化法	22	開発プロセスモデル
5	ブーチ法	14	ダイクストラ法	23	CMM(能力成熟度モデル)
6	コードヨードン法	15	状態遷移モデル	24	三層モデル
7	複合設計法	16	品質機能展開	25	プロジェクトプラン
8	クリーンルーム手法	17	COCOMO		
9	インクリメンタル法	18	COTS		

SEPコースの評価を客観的・定量的に行うために，１年間の修了時点で，成績をOJT（非受講生）の場合と比較することにした．SEPコース受講生とOJTの被験者とで，前提条件が一致するようにした．すなわち，OJTの被験者もSEPコース受講生の選定条件と同一基準により同数（24人）を抽出し，SEPコースと同レベル・同人数の仮想クラスを設定した．そして，SEPコースと同一の事前テストと事後テストを同じ条件で実施した．実施時期も双方同一に行った．

SEPコース受講者とOJT被験者に対して，それぞれ事前テストと事後テストを実施し回帰成就値[47]を求めた．回帰成就値Arは，一般的に次の式で求められる．

$$Ar = S_{post} - (a \cdot S_{pre} + b)$$

ここで，S_{post}およびS_{pre}は，それぞれ事後テストおよび事前テストの得点である．回帰直線は，事前テストから，その学習者が達成できると予測される事後テストの点数を表す．したがって，実際の各個人ごとの事後テスト点数から予測値を差し引いた値が回帰成就値であり，学習効果の指標となる．OJT被験者とSEPコース受講者の学習成績の諸元を表１２．３に示す．

表１２．３からわかるように，OJT被験者とSEPコース受講者は，事前テストの結果はほぼ同じであるが，事後テストでは大きく異なっている．事後テス

トでは，OJT被験者が17.54点に対して，SEPコース受講者は43.33点と，学習成績が大幅に向上していることがわかった．事前テストの影響を考慮して，厳密に成績向上比較を行うために，回帰成就値を算出した．その結果，OJT被験者

表１２．３　OJT被験者とSEPコース受講者の成績諸元

項　　目		OJT	SEP
平均点	事前テスト	10.13	10.38
	事後テスト	17.54	43.33**
標準偏差	事前テスト	8.64	10.47
	事後テスト	7.91	10.86**
回帰成就値	平均	−12.82	12.82**
	標準偏差	2.96	10.00**

（ **：有意水準 1%）

の回帰成就値が−12.82点に対して，SEPコース受講者の回帰成就値が12.82点であった．これらを分かりやすくするため，基点を50点として回帰成就値を加算してグラフ表示したものを図１３．２に示す．

図１２．２から分かるように，OJT被験者が37.2点に対し，SEPコース受講者が62.8点となっており，SEPコース受講者の成績向上度合いが著しい．ｔ検定の結果，OJT被験者に比べてSEPコース受講者の方が1%水準で高い結果であった．

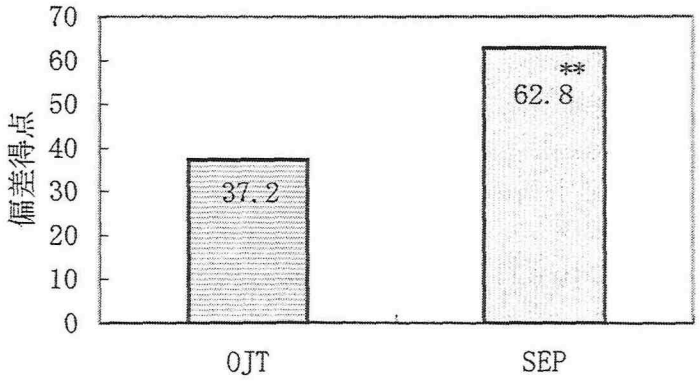


図１２．２　OJTとSEPの学習成績比較（全体）
（**：有意水準1%）

次に，事前テストでの得点が平均点以上の者を成績上位群，平均点未満の者を成績下位群と定義する．成績上位群と成績下位群とで，成績向上度合いがどのようになっているかを，上記の全体と同様に，回帰成就値を算出して比較した．成績上位群について，基点を50点として回帰成就値を加算してグラフ表示したものを図１２．３に示す．

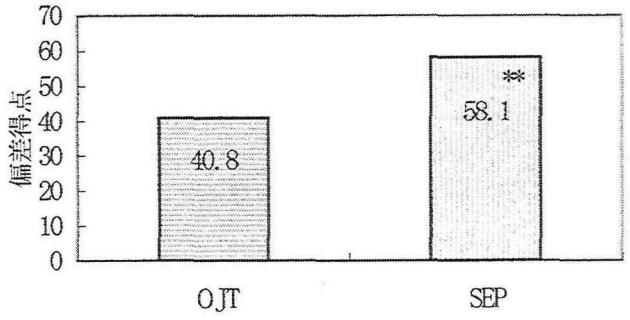


図１２．３　OJTとSEPの学習成績比較（上位群）
（**：有意水準1%）

図１２．３からわかるように，OJT被験者が40.8点に対して，SEPコース受講者は58.1点となっている．ｔ検定の結果，OJT被験者に比べてSEPコース受講者の方が1%水準で高い結果であった．

成績下位群についても同様に，基点を50点として回帰成就値を加算してグラフ表示したものを図１２．４に示す．図１２．４から分かるように，OJT被験者が36.5点に対して，SEPコース受講者は64.4点となっている．ｔ検定の結果，OJT被験者に比べてSEPコース受講者の方が1%水準で高い結果であった．

なお，図１２．３と図１２．４のSEPコース受講者の結果を比較してみると，下位群の方が上位群に比べて，点数が高くなっている．

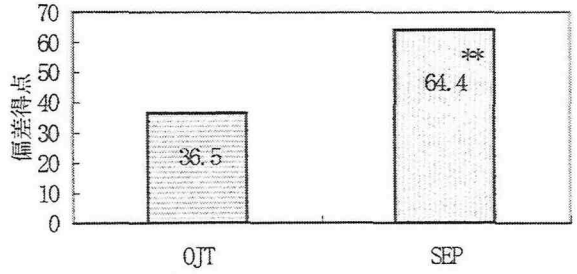


図１２．４　OJTとSEPの学習成績比較（下位群）
（**：有意水準1%）

これは，回帰成就値が上位群，下位群それぞれでの相対的なものであることによるが，SEPコース受講者下位群の成績向上度合いが著しいことを示している．すなわち，SEPコースは下位群に対して成績向上面で非常に有効であることがわかった．

12.4.2 アンケートによる評価

SEPコース受講修了直後，「開発技術」と「管理技術」について調査し，両者を加算した「総合技術」を評価した．全体の93.8%が有用性を認めている．この有用性の内訳を，図１２．５に示す．「実務」とは，修得した技術がそのまま実務に密着していて役に立つことを意味する．「変革」とは，学んだ技術が斬新で現場での業務改革に役立つことを意味する．「知識」とは，今後の設計作業を進める上で必要な知識を身に付けたことを意味する．「有用性なし」とは，上記の有用性が乏しいことを意味し，受講時すでに知識があった人や種々の事情で従来型生産方式を踏襲せざるを得ない人の回答である．

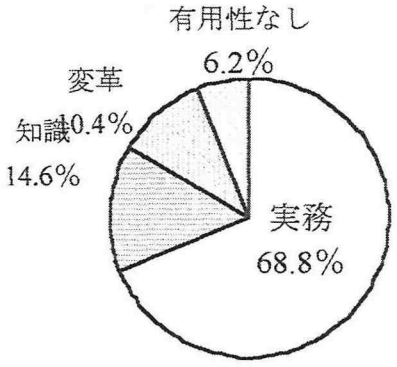


図１２．５　受講直後の有用性調査

次に，受講直後と，受講後２年および３年経過後の有用性評価の推移を図１２．６に示す．

図１２．６から，実務有用性（68.8→54.4→40.4），知識有用性（14.6→13.6→9.8），有用性なし（6.2→4.0→2.8）が減少しているのに対し，変革有用性は，（10.4→27.2→47.0）と増加していることがわかる．これは実務経験を積むことにより，学んだ技術が身に付き，変革意欲が旺盛になるものと解釈している．

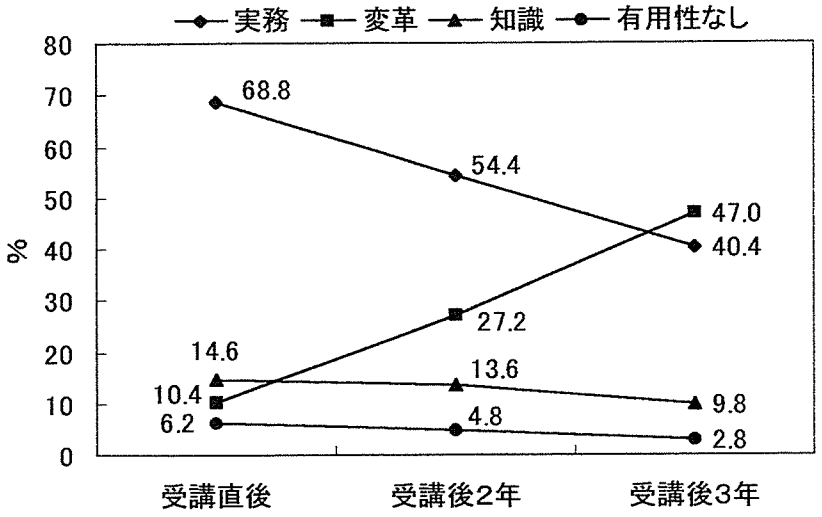


図１２．６　有用性評価の経年変化

12.5 ま と め

企業における上級ソフトウェア技術者育成の実践訓練コースとして，マネジメント・レイヤ，インテグレーティング・レイヤ，エレメント・エンジニアリング・レイヤの３階層からなる教育の概念モデル（図１２．１）を設定した．そして，それぞれの階層における基盤技術となる理論整備の度合いに応じて教育手法を設定するSEPコースを開発し，実践した．その結果，以下のことが明らかになった．

（１）従来型OJT被験者に比べて，SEPコース受講者の方が，上位群，下位群，および全体で成績が向上し，1%水準の有意差があった．
（２）SEPは伸びる人には効果的な学習環境であり，特に下位群に飛躍的な伸びが認められた．
（３）SEPの有用性のうち，変革有用性は受講後の時間経過とともに増加が認められた．

一般的にOJT方式は，指導者の教育への熱意と資質と度量を前提にしている．教育者としても優れた資質のある指導者が期待できる場合に，１対１のOJT方式は威力を発揮することも考えられる．そうでない場合には，教育専門部署によるソフトウェア・ハットを軸とするSEP方式が非常に効果的である．

SEPコースに対して，フェアリ(R. E. Fairley)教授はつぎのように評価して

いる[48].

「SEPは，ソフトウェア・エンジニアリングに対する，おそらく世界でもっとも進んだ産業界の実践訓練コースである．日立製作所の技術研修所は，十分に訓練されたソフトウェア技術者の必要性を認識し，また彼らのゴールを達成するために捧げてきたビジョン，エネルギー，資源に対して，賞賛されるに違いない．SEPの２つの要因が特に重要である．それは，技術面と管理面の両方の強調と，SEPの継続的な改善である．ソフトウェア・エンジニアリングはプロセス重視の学問であるので，技術面と管理面の両方を強調するのは適切である．ソフトウェア・エンジニアリングは急速に変革する学問であり，SEPの継続的な改善は重要である．」

なお，理解度判定については，研修直後だけでなく，一定期間経過後に再度実施し，長期記憶定着を確認する必要がある．これは評価精度を上げるためにも，今後の継続的な研究課題としたい．

第１３章　おわりに

本研究は，1969 年ソフトウェア工場設立と同時に発足した検査部門で筆者が業務を開始して以来 1992 年まで，品質保証と生産技術を直接担当してきた 24 年間の時期に，その基礎がある．この時期は奇しくも日本の高度成長期であり，また情報産業の黎明期から隆盛期に対応している．その後，情報産業も混乱期を迎え，ソフトウェア品質マネジメントも見直しを迫られた．これに対する枠組みの再構築と強化手段の研究が本研究のテーマである．

本研究を推進した筆者の基本的考えは，次の通りである．

・日本のソフトウェア品質マネジメントの弱点を分析する．
・分析結果に基づき，枠組みの再構築の提案を行う．
・日立製作所情報システム部門の具体的強化手段の研究を行う．

第 1 章は，まず研究の主要概念である TQM に対応した「ソフトウェア品質マネジメント」を説明し，本研究の背景と目的および本論文の構成を概説した．

第 2 章から第 4 章では，日本型ソフトウェア開発の混乱の原因と対策方針を示した．

第 2 章は，欧米のソフトウェア開発が研究所スタイルであるのに対して，日本で 1980 年代に確立した，従来の製造業をモデルにした「ソフトウェア工場」開発方式，およびそれを基盤としたソフトウェア品質管理の概要と特徴を整理した．

第 3 章は，まずこれまでの日本のソフトウェア品質管理の変遷を整理し，次いで１９９０年代日本におけるソフトウェア開発の問題点の構造を明らかにした．すなわち，変化への対応が弱い経営体質，ＩＳＯ９０００対応はしたものの，依然としてグローバル化が不十分な開発体制，リスク管理が出来ていない QCD 主体プロジェクト管理，ユーザニーズの把握とそれに基づく仕様決定力が弱い開発技術力という実態が浮かび上がった．これを踏まえて，技術面，プロジェクト管理，品質戦略という側面から各々課題と対策案を検討した．

第 4 章では，第 3 章の「1990 年代ソフトウェア品質マネジメントの課題と取り組み方」をもとに，「ソフトウェア品質マネジメントの枠組みの再構築と対策方針」を示した．

第 5 章から第 12 章では，第 4 章の対策方針の各項目に対応して，具体的な強化策を詳細に記した．

第5章は，グローバルスタンダードと整合する品質マネジメントシステムを示し，さらにこれを実現するための組織（プロジェクトオフィス機能を従来組織に適合するようにして導入）と仕組みを提案した．

第6章と第7章は，従来から日本のソフトウェア開発の弱点と言われてきた上流工程に対する対策である．

第6章は，従来のＱＣＤ（品質，コスト，納期）中心であったプロジェクトマネジメントに，６つの知識領域を加えたモダンプロジェクトマネジメントを採用することで，特に上流工程のリスク管理を強化する施策（リスクマネジメント事前自己診断シート）を中心に述べた．

第7章は，仕様の早期明確化と仕様変更に対する施策である．これはＱＦＤ（品質機能展開）という品質管理手法を応用した「仕様発散防止ＱＦＤ」を考案して，仕様確定状況の監視支援を行うというものである．

第８章は，レビュー品質向上手段を開発・導入することにより設計品質の向上を図るという施策である．各レビュー時に作成する議事録をデータベース化し，検討項目ごとに編集して仕様決定の経過を分かりやすく示すとか，標準レビューテーマと実際の検討内容を比較して検討不十分な項目をチェックできるようにするものである．

第９章は，プロジェクト管理の精度向上を図るために，「プロジェクトナビゲーションツール」を開発し，適用するという施策である．このツールは，ＷＢＳの標準化，成果物の一元管理と再利用，およびＷｅｂ／グループウェア技術の利用による分散開発の高効率化を狙いとしている．例えば，プロジェクト管理者にとっては，開発者からの特別な報告がなくても状況把握ができるとか，成果物の現物確認が容易である等の効果がある．

第 10 章は，テスト段階の品質評価精度の向上と効率的なフィードバックを可能にする手法を提供する．具体的には，複数の指標の相関評価とそのパターン化や，評価要素の目標値達成度による評価点法の導入により，比較的簡単に品質評価ができる．

第 11 章は，マルチベンダ構成の情報システム向けシステムシミュレーションテスト（ＳＳＴ）の導入である．ＳＳＴとは，それぞれ検査合格した製品を自社内のセンタに持ち込み，出荷前に顧客システムを擬似した環境下で行うテストである．自社製品主体のＳＳＴは 20 年以上から実施しており，効果をあげてきたが，近年マルチベンダ構成や多岐にわたるネットワーク網を利用した情報システ

ムが増加し，従来型ＳＳＴでは実現が困難になってきた．そこで，他社製品やネットワーク網の投資と共にテストノウハウを蓄積し，マルチベンダ構成ＳＳＴを実現した．

以上第 6 章から第 11 章は，いずれも開発プロセスの改善について述べた内容であるが，次の第 12 章は人材育成に関するものである．これは，ソフトウェア開発リーダ向けの開発技術と管理の両面を，講義とミニプロジェクトでのグループ演習を通じて習得する長期研修である．これは 1990 年から開始し，改良を加えてきた．この評価を試みた．

以上をまとめると，本研究による研究成果は，

・20 世紀最後の 10 年間のソフトウェア開発混乱の原因分析と対策指針の提示
・21 世紀初頭のソフトウェア品質マネジメントの枠組みの提案
・対策指針に対応する，各プロセスごとの改善施策の実現と適用評価
・ソフトウェア開発リーダ向け人材育成研修の推進と評価

である．これらは，1990 年代の混乱のなかで，同時並行的に試行錯誤を繰り返しながら，多くの関係者の努力により，達成されたものである．本研究は，そうした成果のなかから，21 世紀初頭のこの 2 年間に，筆者と当該関係者が精力的にまとめて公表した研究結果を，筆者の責任で体系化したものである．

研究テーマが大きく，多岐にわたるため，これらの総合的評価を定量的に述べるのは困難である．しかしながら，一時は先の展望が見えず混乱した開発現場でも，現時点で問題がなくなったわけではないが，各施策の適用状況は着実に増加しており，対策方針に対する信頼感は増している．

次に，今後の課題と展望を述べる．

本研究で対象としたのは，主として情報システムを構成するソフトウェア開発における品質問題である．今後の課題としては，大きく３分野がある．第１は，情報サービスの品質問題である．第２は，情報システムの構築ではなく，情報システムの提案力の問題である．第３は，組込みソフトウェアの開発力と品質である．

（1）情報サービスの品質問題

ソフトウェア開発という観点でみると，情報システム分野では，従来のような自社の技術と製品だけで，ゼロから大規模システムを長期間にわたって開発するという事例が最近極めて少なくなっており，できる限り各ベンダで提供するソフトウェアパッケージを利用して短期開発するというのが主流である．この分野の傾向としては，ソフトウェア開発というよりもサービスビジネスが増加しつつある．従来の情報システムは，業務のコンピュータ化や合理化が主体であったが，

現在は業務についていえば，例えば自社内での合理化よりも業務自体のアウトソーシングによる経営の合理化が狙いであり，情報産業側からみれば，これはアウトソーシングビジネスの提案であり，そのサービスビジネスの遂行である．ここでの品質問題は，例えば顧客を満足させられるビジネスモデルの提案の質や，受注したサービスの品質をどう定義し，どう評価するかである．この１例として，SLM（サービスレベルマネジメント）がある．これはサービスの品質を管理するものである．しかし，サービスビジネスの対象は広範囲にわたっており，サービス品質に対する取組みについては，まだ研究の緒についた段階であり，解決すべき課題は山積している．

（２）情報システムの提案力の質の問題

情報システム分野の現在の最大関心事は，定常業務ではなく，顧客の事業のあり方，経営のあり方そのものに対する提案やコンサルテーションの質である．従来の上級ＳＥに求められたスキルは，顧客業務を理解して情報システムの要求仕様をまとめることであったが，現在および今後の上級 SE には，上記のように，顧客の経営課題自体に対する提案力やコンサルテーション力が要求されている．この分野では，日立製作所でも上級ＳＥ教育に着手しているが，品質問題として具体的に展開するまでには至っていない．

ソフトウェア品質マネジメントという研究テーマと，顧客経営に対する提案とは一見かけ離れているようにみえるが，経営の変革が迫られている現代には密接なつながりがあるということである．これは，日本の品質管理が QC，TQC,そして TQM へと拡大進化してきたことと対応している．

（３）組込みソフトウェアの品質問題

現在急速に伸びているのが,ハードウェア製品に内蔵されているソフトウェア，いわゆる「組込みソフトウェア（embedded software)」である．携帯電話各社の新機種が次々とソフトウェアトラブルで回収騒ぎになったのは，まだ記憶に新しいが，様々な分野の多くの新製品開発において，組込みソフトウェアが隘路となりつつある．ここでの大きな問題点は２つある．１つは，ソフトウェア開発の管理者と技術者の量的，質的の両面での深刻な不足である．第１２章で取上げた上級ソフトウェア技術者実践教育ＳＥＰでも，１０年前は情報系受講者が主体であったが，この数年は組込みソフトウェア系受講者が主体となりつつある．２つめは，ハードウェア製品主体の開発製造体制のなかで，如何に適切な組込みソフトウェア開発の体制と仕組みを考案し，導入・定着化するかである．ハードウェア開発と共存する組込みソフトウェア開発のフレームワークの構築が課題である．

参考文献

[1] 保田勝通：ソフトウェア品質保証の考え方と実際,日科技連出版社，1995

[2] 石川馨：日本的品質管理，日科技連出版社，1981

[3] 東基衛(編)：ソフトウェア品質評価ガイドブック,日本規格協会，1994

[4] M.A.Cusumano: Japan's Software Factories, Oxford University Press, 1991（富沢他(訳)：日本のソフトウェア戦略(Japan's Software Factories),三田出版会,1993)

[5] プロジェクトマネジメント学会(編)："特集エンタープライズプロジェクトマネジメント"，プロジェクトマネジメント学会誌，Vol.3, No.5, 2001

[6] 保田勝通："ソフトウェアの信頼性向上施策と標準化動向"，品質管理，Vol.50, No. 7, pp.664-673, 1999

[7] F.P.Brooks, Jr.: The Mythical Man-Month: Essays on Software Engineering (Anniversary Ed.), Addison-Wesley Publishing, 1995（滝沢他(訳)：人月の神話，アジソンウェスレイ，1996)

[8] 富士通通信ソフトウェア開発部編：富士通における「あゆみ」活動－高品質ソフトウェア開発への挑戦－,日科技連出版社,1992

[9] PMI: A Guide to the Project Management Body of Knowledge(PMBOK), Project Management Institute, Pennsylvania,2000（エンジニアリング振興協会(訳)：プロジェクトマネジメントの基礎知識体系(PMBOK Guide 和訳版)，エンジニアリング振興協会，1997)

[10] 中村翰太郎(編)，エンジニアリング振興協会(監修)：＜対訳＞ISO/JIS Q 10006 －品質マネジメントー プロジェクトマネジメントにおける品質の指針とその解説，日本規格協会，1998

[11] 松原友夫："ベストプラクティスの思想と実践"，情報処理，Vol.39, No.1, pp.43-48,1998

[12] E. ヨードン(著)，松原友夫(訳)：プログラマーの復権，トッパン，1997

[13] DoD Software Acquisition Best Practice Initiative: The Program Manager's Guide to Software Acquisition Best Practices (Version 2.1), Software Program Managers Network, 1998. http://www.spmn.com

[14] 飯塚悦功(監)：ソフトウェア ISO 9000, 日科技連出版社, 1996

[15] 青山幹雄："ソフトウェア技術者のグローバルスタンダード化"，情報処理，Vol.39, No.11, pp.44-47,1998

[16] 日本工業標準調査会(審議)：日本工業規格 ソフトウエアライフサイクルプロセス JIS X 0160(ISO/IEC 12207), 日本規格協会，1996

[17] SLCP-JCF98 委員会(編)：共通フレーム９８ SLCP-JCF98, 通産資料調査会，1998

[18] M. Paulk and et al.: The Capability Maturity Model, Addisson-Wesley,

1994（アンダーセンコンサルティング（監訳）：成功するソフトウエア開発ーCMMによるガイドライン，オーム社，1998）
[19] 保田勝通:"20 世紀のソフトウェア品質管理と 21 世紀の課題"，品質管理，Vol. 51，No.12，pp.1085-1097, 2000
[20] 金子龍三：ソフトウェア開発体質改革論，日科技連出版社，1999
[21] E．ヨードン（著），松原友夫，山浦恒央（訳）：デスマーチ，トッパン，1998
[22] G. M. ワインバーグ(著)，大野侑朗(監訳)：ワインバーグのシステム思考法，共立出版，1994
[23] 日本工業標準調査会（審議）：品質マネジメントシステムーパフォーマンス改善の指針 JIS Q 9004:2000(ISO 9004:2000)，日本規格協会，2000
[24] 水野滋，赤尾洋二(編)：品質機能展開，日科技連出版社，1997
[25] 赤尾洋二(編)：品質展開活用の実際，日本規格協会，1988
[26] 情報処理振興事業協会センタ(編)：品質機能展開による高品質ソフトウェアの開発手法，コンピュータ・エージ社，1989
[27] 堀内純孝:役に立つデザインレビュー，日科技連出版社，1992
[28] 菅野文友：ソフトウエア・プロジェクト管理(下)，ソフト・リサーチ・センター，1990
[29] Y. Kudo, C. Hirai, Y. Furuhata, and et al. : "A Proposal of a Review-Report-Oriented Knowledge-Management Model", Proc. 2nd World Congress for Software Quality, pp.199-204, 2000.
[30] N. Yoshida and A. Ooishi: "Improvement in the Quality Evaluation Precision of the Software and an Efficient Feedback Technique", Proc. 2nd World Congress for Software Quality, pp.351-356, 2000
[31] 喜多村直矢，吉田直美他："超幾何分布モデルの適用性評価およびソフトウェア信頼度モデルを使用した稼動後の不良発生予測"，第 18 回ソフトウェア生産における品質管理シンポジウム発表報文集，pp.209-216, 1998
[32]山田茂：ソフトウェア信頼性モデルー基礎と応用ー，日科技連出版社，1994
[33] K. Yasuda and K. Koga: "Product development and quality assurance in the software factory", in Software quality assurance and Measurement World Perspective (Eds. N. Fenton, R. Whitty, and Y. Iizuka), International Computer Press, 1995
[34] K. Yasuda: "Software Quality Assurance Activities in Japan", in Japanese Perspectives in Software Engineering (Eds. Y.Matsumoto and Y.Ohno), Addison-Wesley, 1989
[35] S.Moriguchi:Software Excellence - A Total Quality Management Guide, Productivity Press,1997
[36] S. Uryu and M. Shimizu:"System Simulation Testing: Pre-Delivery Testing to Ensure System Dependability", Proc. 4th Intern. Symp. Software Reliability Engineering, pp. 208-217, 1993
[37] P. B. Crosby: Quality Is Free, McGraw-Hill, 1979

[38] 石井康雄，高橋延匡，飯塚悦功，菅野文友(編)，保田勝通他(著)：ソフトウェアちょっといい話'93ユーザのニーズに応える最新技術，日科技連出版社，pp.41-57,1993
[39] 大槻 繁，伊藤泰樹，角 行之，保田勝通他："知識主導社会へ向けての上級ソフトウェア技術教育"，工学教育 第47巻，第4号，pp.8-13,1999
[40] R. Dawson and R. Newsham:"Introducing Software Engineers to the Real World", IEEE Software,Vol.14,No.6, pp.37-43,1997
[41] 吉田幸二，内田雅幸，小泉寿男："企業から大学教育への取組について" 信学技報 ET96-75, 1996
[42] P. Bourque et al. :Guide to the Software Engineering Body of Knowledge A Strawman Version, 1998
[43] 情報処理学会 ：情報システム学とソフトウェア工学のカリキュラムに関する調査研究報告書，2000
[44] IEEE :IEEE Software Engineering Standards Collection 1999, IEEE Computer Society Press, 1999
[45] R. H. Thayer (ed.): Tutorial: Software Engineering Project Management, IEEE Computer Society Press, 1988
[46] ACM :IS'97 Model Curriculum and Guide Lines for Undergraduate Degree Programs in Information Systems, 1997
[47] 池田 央：テストと測定，第一法規出版，1982
[48] S. Otsuki, S. Mitsumori, and T.Kado :"Challenge to Software Hut Education: The Effectiveness of Software Engineering Project Education", Proc. IFIP/SEARCC World Conference, pp.99-108, 1993
[49] R. E. Fairley :"Guest Editor's Introduction to Special Issue on Software Engineering Education", IEEE Trans. Software Engineering, Vol. SE-13, No.11, pp.1141-1148, 1987

謝辞

本研究を遂行するにあたり，多大のご指導・ご鞭撻を戴きました鳥取大学工学部社会開発システム学科山田茂教授，河合一教授，および知能情報工学科池原悟教授に深く感謝いたします．特に指導教官の山田茂教授には鳥取大学大学院工学研究科博士課程に在学中，懇切なご指導と格別のご配慮，暖かい励ましを賜り，心から感謝しております．

本研究は，日立製作所入社以来当時のソフトウェア工場検査部および設計部，さらに情報システム工場での検査部と生産技術部での業務が基礎になっており，この成果をまとめた著書「ソフトウェア品質保証の考え方と実際」が本研究のスタートになっております．その当時の上長や同僚他，関係者各位に改めて感謝の意を表します．

その後，同技術研修所での研修企画業務の傍ら，上記部門との交流に加えて，日本科学技術連盟のソフトウェア生産管理研究委員会（ＳＰＣ委員会）や，標準化活動の各種委員会，日本信頼性学会，プロジェクトマネジメント学会等社外の活動を通じて多くのことを学ばせて頂きました．それらが本研究のバックグラウンドになっております．これら社外活動を通じてご指導いただいた多くの諸先輩や活動を共にした委員の皆様方にこの場を借りて感謝します．また，こうした活動を快く支援して頂いた同技術研修所の歴代所長と長島現所長に感謝致します．

本研究の第６章以降は，この２年間の日立製作所情報グループ生産技術統括センタ大野治センタ長，同生産技術本部原田晃本部長，同プロジェクトマネジメントセンタ高橋センタ長他の皆様や同品質保証統括センタ毛利センタ長他の皆様の絶大なご協力によるところが大きく，改めて謝意を表します．

最後に，本研究を進めるにあたり日ごろから心の支えとなり，また多くの支援をしてくれた妻紀代子に感謝します．

研究業績一覧表

主論文

[1] 保田勝通，大石晃裕，吉田直美，山田茂："グローバル時代のソフトウェア品質マネジメントシステムの提案と試行"，プロジェクトマネジメント学会誌，Vol.3, No.2, pp.21-26, 2001年4月

[2] 角行之，保田勝通，山本洋雄，大槻繁："企業における上級ソフトウェア技術者養成のための実践訓練コースＳＥＰの開発と評価"，教育システム情報学会誌，Vol.18, No.1(春号)，pp.111-120, 2001年4月

[3] K. Yasuda, K. Tokunaga, M. Shimizu, and S. Yamada: "Measures to Increase the Dependability of Information Systems in the IT Age: A System Simulation Testing Approach", Supplemental Proceedings of the 12th International Symposium on Software Reliability Engineering (ISSRE 2001), pp.174-184, 2001年11月

[4] M. Nakata and K. Yasuda: "Quality Assurance Activities for ASP Based on SLM in Hitachi", Proceedings of the 7th European Conference on Software Quality (ECSQ 2002) , pp.82-89, 2002年6月

[5] Y. Furuhata, Y. Makuta, H. Komuro, O. Ohno, A. Harada, Y. Kudo, and K.Yasuda: "A Project Management Tool Utilizing Review Reports for Software Development", Proceedings of the International Conference on Project Management (ProMAC 2002), pp.171-176, 2002年7月

[6] A. Harada, O. Ohno, K. Uehara, H. Komuro, K. Fujii, and K.Yasuda: "Project management with "Pro-Navi"", Proceedings of the International Conference on Project Management (ProMAC 2002), pp.73-79, 2002年7月

[7] A. Harada, O. Ohno, K. Uehara, H. Komuro, M. Kurashige, and K. Yasuda: "Specification Instability Prevention QFD - A Practical Application of QFD for Business Custom Software Project", Proceedings of International Conference on Project Management(ProMAC 2002), pp.319-322, 2002年7月

[8] T. Minamino, H. Toyama, and K. Yasuda: "An Application of Modern Project Management to "IT" System Development Projects", Proceedings of International Conference on Project Management (ProMAC 2002), pp.323-327, 2002年7月

[9] H. Komuro, O. Ohno, Y. Furuhata, A. Harada, Y. Makuta, and K. Yasuda : "Improving an Accuracy of Estimation in a Software Development with a Specific Program Development Automation", Proceedings of International Conference on Project Management (ProMAC 2002), pp.461-470, 2002年7月

[10] K. Yasuda and S. Yamada: "The Concept and Practice of Software Quality Assurance in Japan in the Global Era", Proceeding of the 7th IEEE/IEICE International Conference on High Assurance System Engineering (HASE 2002), pp.13-18, 2002年10月

参考論文（著書）

[1] 保田勝通（単著）：ソフトウェア品質保証の考え方と実際，日科技連出版社，1995年11月,全436頁

[2] 保田勝通：第3章プログラム仕様の表現法とプログラム仕様書，ソフトウェアの製造－日科技連ソフトウェア品質管理シリーズ3－，石井康雄(編)，日科技連出版社出版社，pp.85-139，1986年12月

[3] 保田勝通：第7章品質保証プログラム推進事例，ソフトウェアの品質管理と生産技術，吉澤，東他(編)，日本規格協会，pp.129-147，1988年10月

[4] K. Yasuda: Chapter 8 Software Quality Assurance Activities in Japan, Japanese Perspectives in Software Engineering (Eds. Y. Matumoto and Y. Ohno), Addison-Wesley Publishing, pp.187-205, 1989年

[5] 保田勝通：第12章品質の定量的計測と評価の方法（12.7節，7頁），第13章品質設計と管理計画（13.8節，3頁），第17章テスト管理と検査（17.2，17.3節，7頁），第23章品質保証推進事例，ソフトウェア品質管理ガイドブック，森口繁一(編)，日本規格協会，pp.244-247，pp.279-281，pp.345-352，pp.467-473, 1990年7月

[6] 保田勝通:「第3章DIS 9000-3対訳と解説」の規格の章節「5．5 設計と製造」，ソフトウェアの品質保証－ISO/DIS 9000-3対訳と解説，飯塚悦功(編)，日本規格協会, pp.66-69,1990年7月

[7] 保田勝通：解説編 2.9 ソフトウェア品質管理の考え方事例編，C7 品質保証技術／管理技術事例８５－８６，：ソフトウェア品質管理事例集，日科技連ソフトウェア生産管理運営委員会（編），日科技連出版社，pp.49-59，pp.1035-1058, 1990年10月

[8] 保田勝通：第Ⅰ部「ユーザのニーズに応える最新技術」発表論文3「ソフトウェア開発の標準化動向とその影響」，ソフトウェアちょっといい話'93 ユーザのニーズに応える最新技術，石井,高橋，飯塚,菅野(編)，日科技連出版社，pp.41-57, 1993年9月

[9] K. Yasuda: Chapter 18 Product Development and Quality Assurance in the Software Factory, Software Quality Assurance and Measurement – A

Worldwide Perspective (Eds. N. Fenton,Y. Iizuka, and et al.), International Thomson Computer Press, pp.195-205, 1995年

[10] 保田勝通：第４章ＩＳＯ　９００１の各要求事項のソフトウェアへの適用(4.1-4.2)，ソフトウェアＩＳＯ　９０００，飯塚悦功(監修)，日科技連出版社，pp.59-75，　1996年11月

[11] K. Yasuda: 13.7 Evaluation efficiency,　18.2 Software Test Planning and Test Design, 18.3 Software Test Implementation Management, Software Excellence - A Total Quality Management Guide (Eds. S. Moriguchi), Productivity Press, pp.367-372,　pp.522-530, pp.530-533, 1997年4月

[12] 保田勝通：第Ｖ部信頼性・保全性・安全性の事例Ｄ－８ ソフトウェア製品の品質保証，信頼性ハンドブック，日本信頼性学会（編），日科技連出版社 pp.851-859，1997年4月

参考論文（研究報告・発表・解説）

[1] 保田勝通，味松康夫："高信頼性システムを実現するための専用設備 SST"，日立評論，Vol.61,No.12,pp.61-64,1979 年 12 月

[2] 保田勝通："ソフトウェアのテスト技術と品質評価"，品質(日本品質管理学会誌), Vol.12,No.2,pp. 65-70,1982 年 2 月

[3] 保田勝通，小国力，片岡雅憲："ソフトウェア生産技術の現状と将来特集／３．５保守ツール"，電子通信学会誌, Vol.66, No.4, pp.378 –383,1983 年 4 月

[4] 保田勝通，野木兼六，古川善吾："ソフトウェアテスト項目作成支援システム"，日立評論，Vol.66, No.3, pp.29-32, 1984 年 3 月

[5] 保田勝通:"テスト・品質保証技術の現状と課題", 情報処理(情報処理学会誌), Vol.28, No.7, pp.873-879, 1987 年 7 月

[6] 保田勝通:"ソフトウェアの品質保証技術", 電子情報通信学会誌, Vol.73, No.5, pp.467-474, 1990 年 5 月

[7] 保田勝通，東基衛:"ソフトウェアの品質定量評価とテスト管理", 情報処理 (情報処理学会誌)，Vol.33, No.8, pp.934-944,1992 年 8 月

[8] 保田勝通，堀田文明，香村求，大筆豊，野木秀子，松尾谷徹："特集 ソフトウェアの品質保証について：産業界から学会へのメッセージ"，品質（日本品質管理学会誌)，Vol.23, No. 2 , pp.169-173,1993 年 2 月

[9] 角行之，大槻 繁，坂口晴一郎，山崎重之，保田勝通，伊藤泰樹："知識主導社会へ向けての上級ソフトウェア技術教育"，工学教育，第 47 巻，4 号，pp.8-13,1999 年 7 月

[10] 保田勝通:"ソフトウェアの信頼性向上施策と標準化動向"，品質管理, Vol.50, No.7, pp. 664-673, 1999 年 7 月

[11] 保田勝通："20 世紀のソフトウェア品質管理と 21 世紀の課題"，品質管理, Vol.51, No.12, pp. 1085-1097, 2000 年 12 月

END